GIS 4653/5653: Spatial Programming and GIS

Automating GIS tasks in ArcGIS

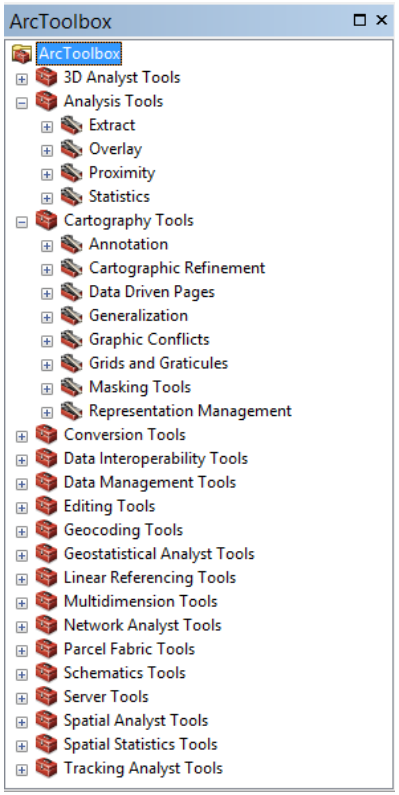
Interactive Geoprocessing

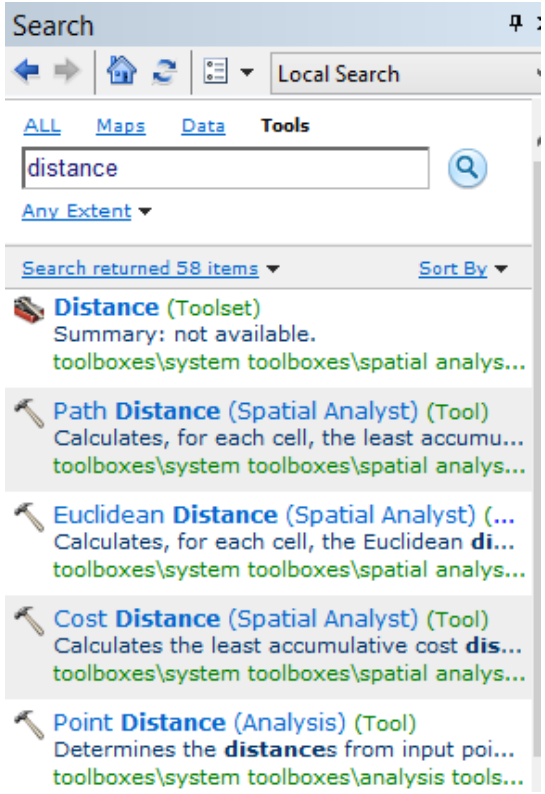
Automating GIS tasks in ArcGIS

- Many of the interactive operations in the ArcGIS GUI are merely front-ends to geoprocessing tools
  - Can invoke these tools directly without going through the ArcMap graphical interface
- Two ways to invoke geoprocessing tools
  - Model Builder
  - Python scripts

Geoprocessing tools

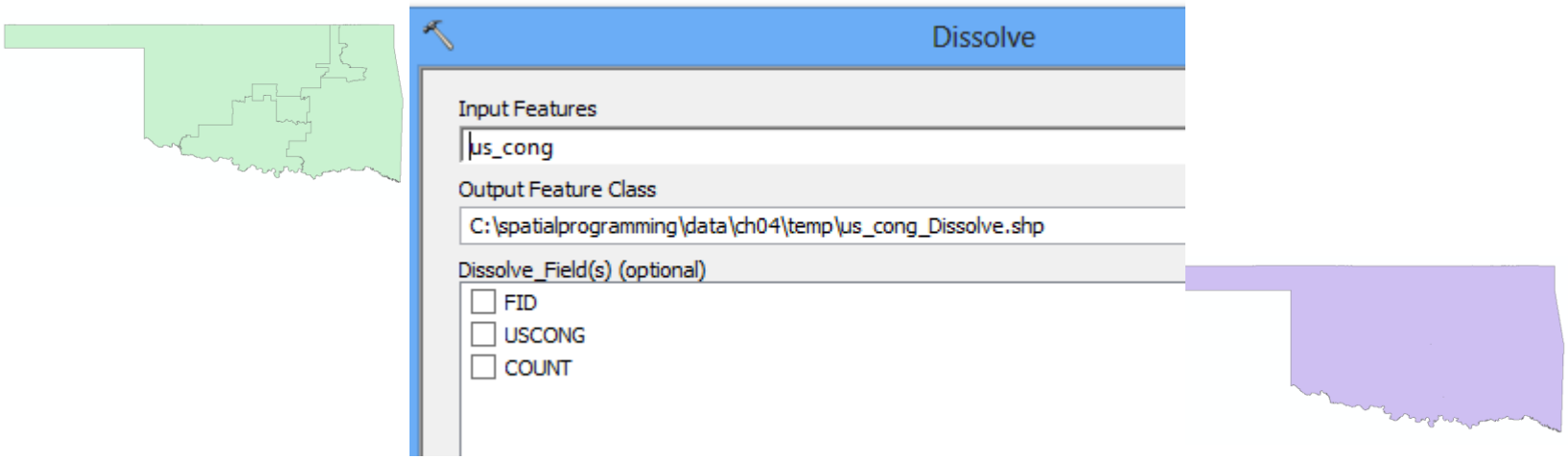
- The basic geoprocessing tools are available directly
  - Many more are in the ArcToolbox and can be found by Search





Interactive use with the dialog

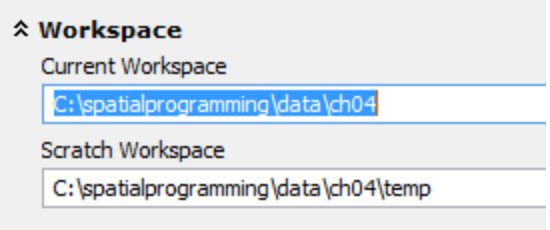
- Let us take the example of dissolving the congressional boundaries to create a state boundary file of Oklahoma
- Geoprocessing | Dissolve | Fill dialog box



- To avoid errors, choose input feature from display layers

Environment options

- When carrying out geoprocessing operations, it is good to change a couple of ArcMap environment settings
  - Specify the current workspace explicitly
    - Geoprocessing | Environments | First option
    - Either the location of your script or the location of your data



  - Allow scripts to overwrite earlier outputs
    - Changes from being an error to merely a warning
    - Geoprocessing | Geoprocessing Options | First check box

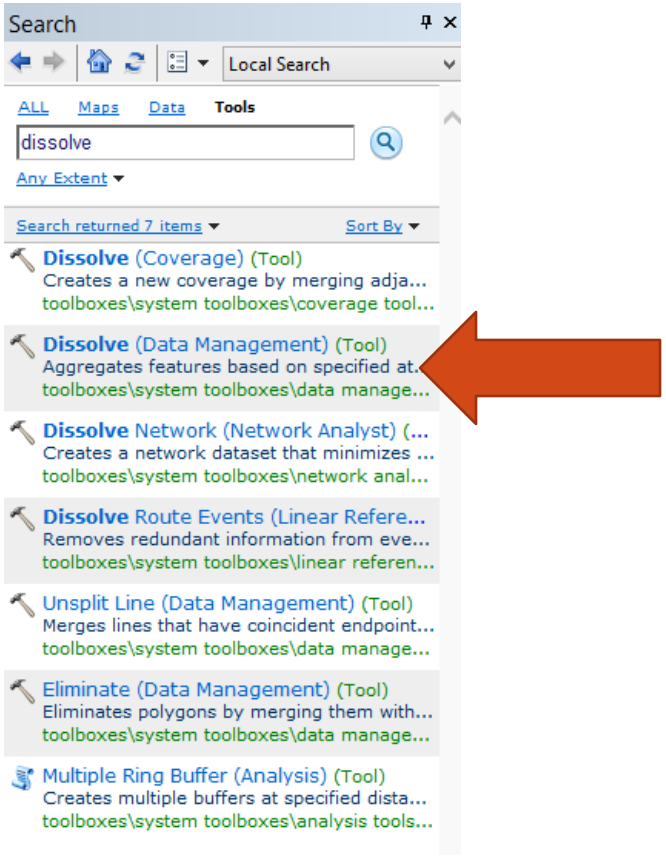
Hierarchy of Environment Settings

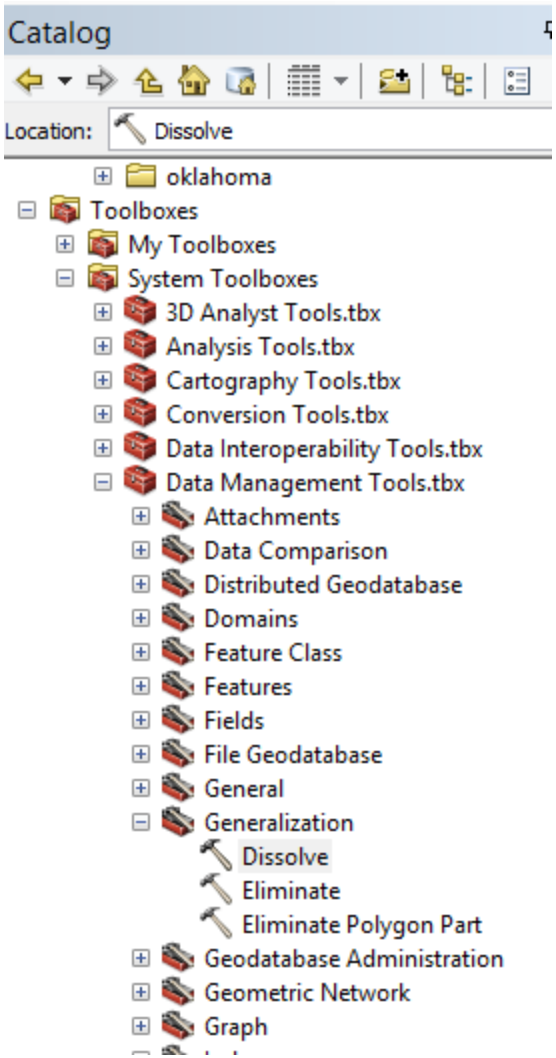
- The most specific over-rides the more general
- From most general to most specific:
  - ArcMap
  - Individual tool
    - Click on "Environments" on the dialog box to change
  - Model
    - Environment settings that are part of the model properties
  - Python scripts can set environment variables

Batch processing

- Can use ArcMap interactively if the same tool needs to be applied to many datasets
- To bring up the batch window, right-click on the tool name in the Catalog window
- Which toolbox is "Dissolve" in?

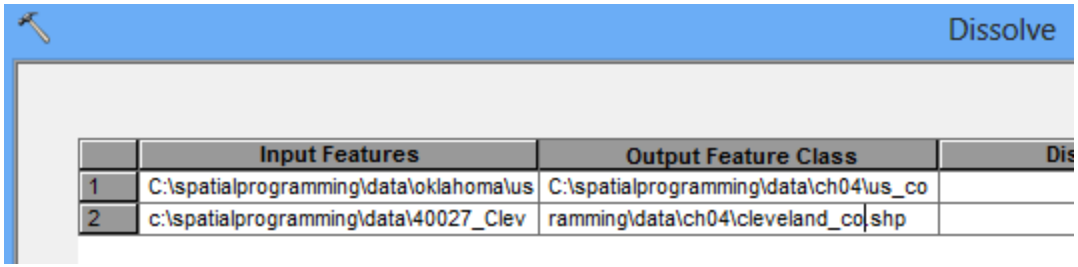
Use Search to find tool





Batch window for Dissolve

- Specify a set of parameters
  - Use copy-paste from one line to the next
  - Can also drag and drop features from active map
  - Right-click in a textbox to obtain a Browse window



  - Can have the tool check for errors

Model Builder

Using a Model Builder

- A model in ArcGIS is a workflow built using a visual programming environment
  - Useful when you tend to keep running a sequence of tools over and over again
  - Can use loops and conditions to control the workflow

Elements of a Model

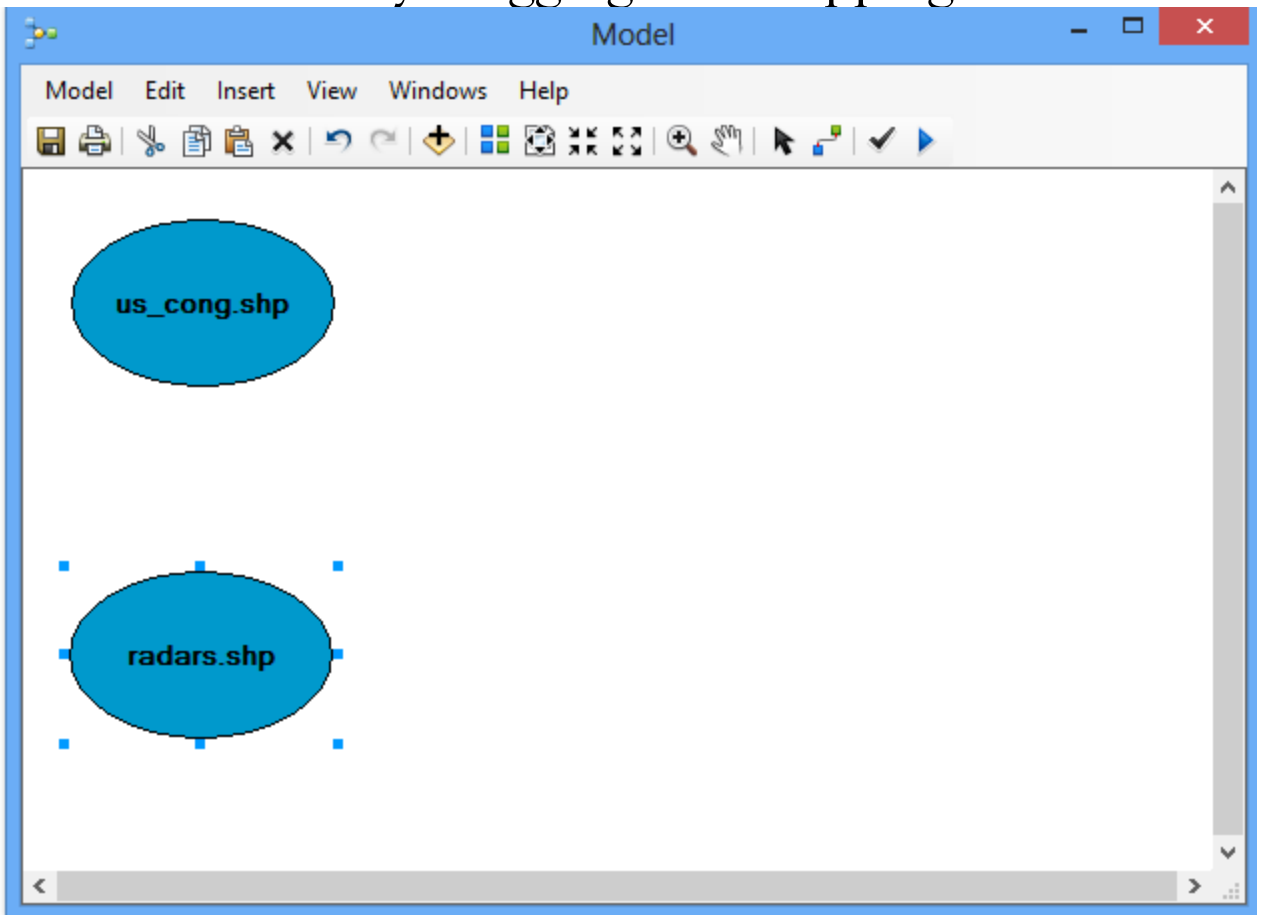
- A model consists of:
  - Input data
    - This is typically a layer either in a shapefile or in ArcMap
  - Tool
    - Geoprocessing tasks that operate on data
  - Value
    - Inputs other than data
    - Strings, numbers, etc.
    - Environment settings
  - Derived data
    - Intermediate or final output of tools in the Model

A simple model

- Let us build a model that will:
  - Dissolve a given shapefile to the outer boundary
  - Buffer the boundary by 50 km
  - Clip the shapes in a second shapefile to the dissolved boundary of the first
- What is the:
  - Input Data?
  - Tool?
  - Variable?
  - Derived data?

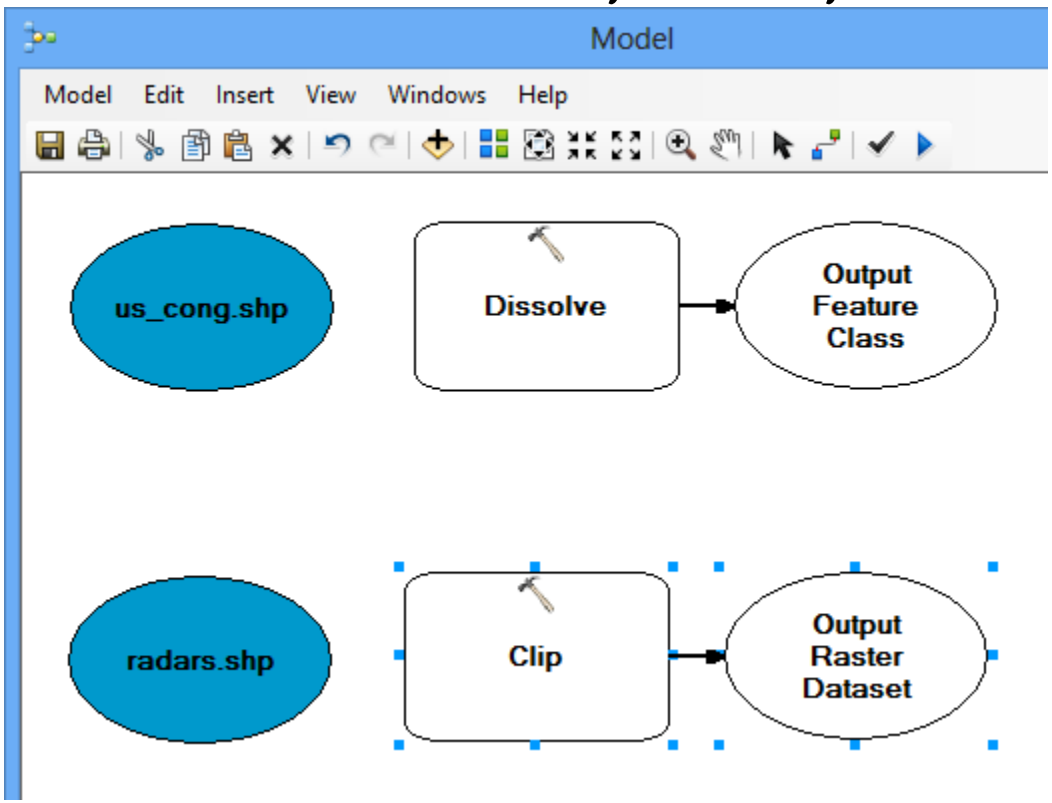
Specifying the model

- Select Geoprocessing | ModelBuilder
  - Add data to model by dragging and dropping



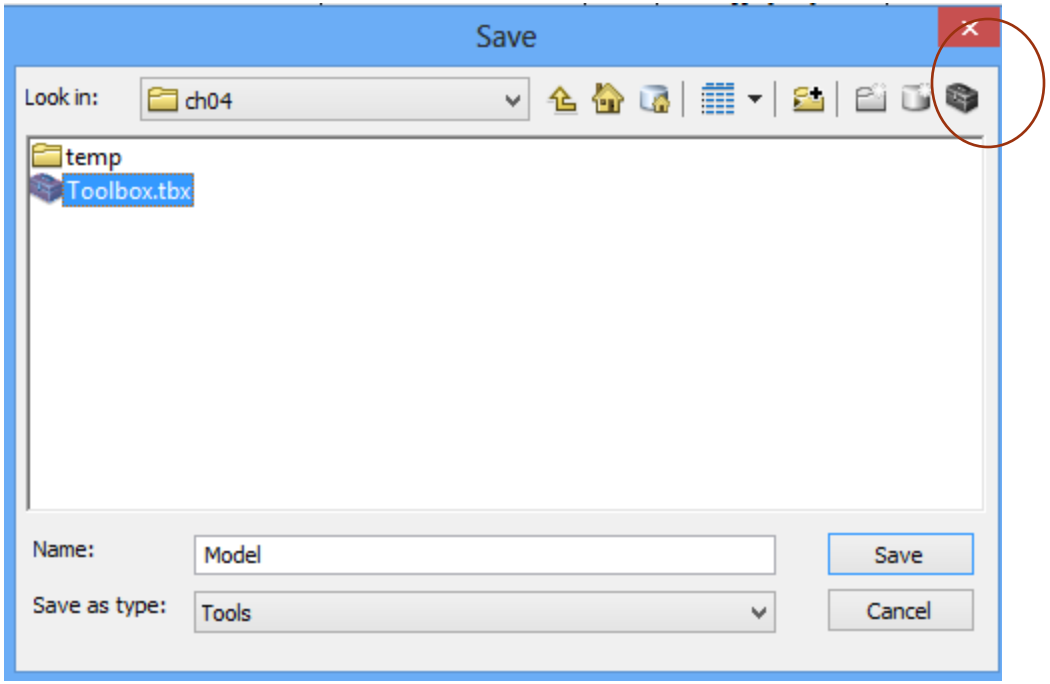
Adding tools

- Drag and drop tools from the toolbox menu
  - The tools are white because they are not yet connected



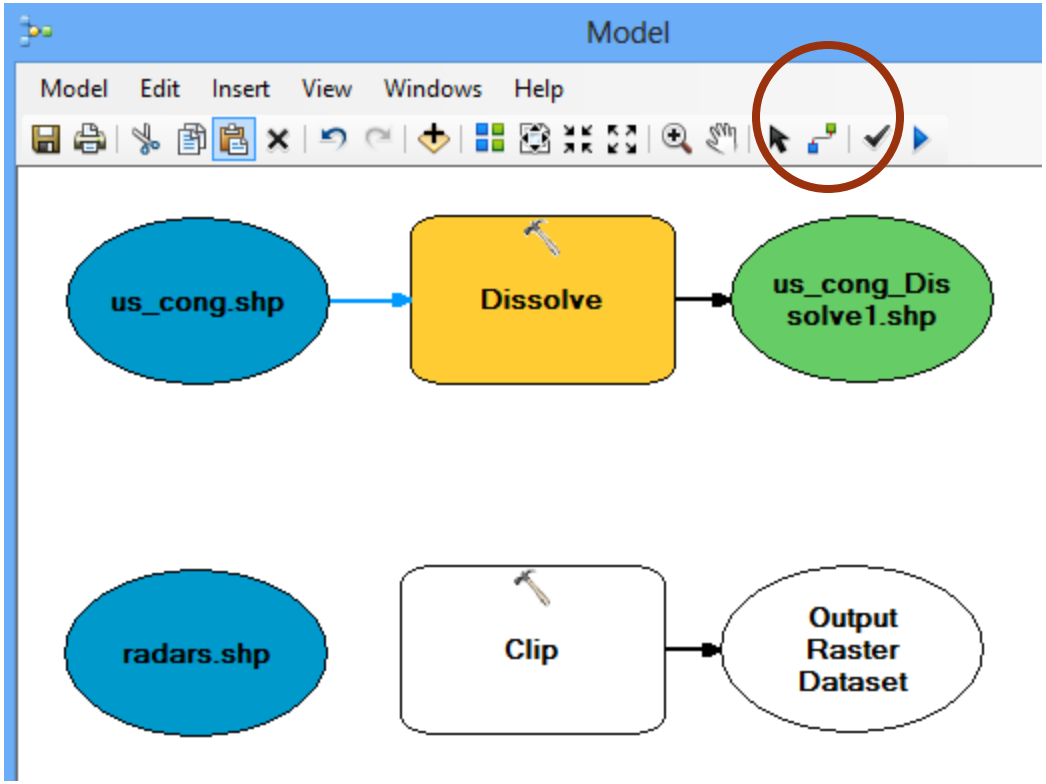
Saving a model

- A model can only be saved within a toolbox
  - Make a new toolbox to save models



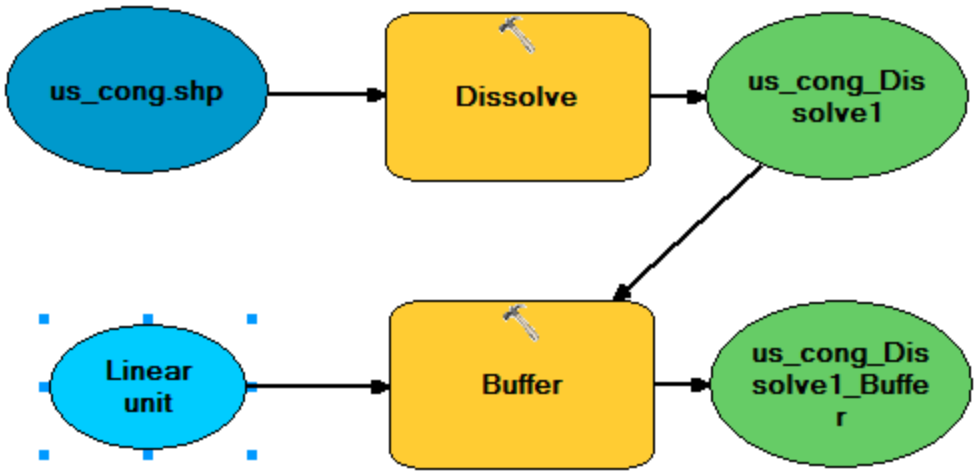
Draw connectors

- When you draw the connector, ArcMap suggests what type of input it could be

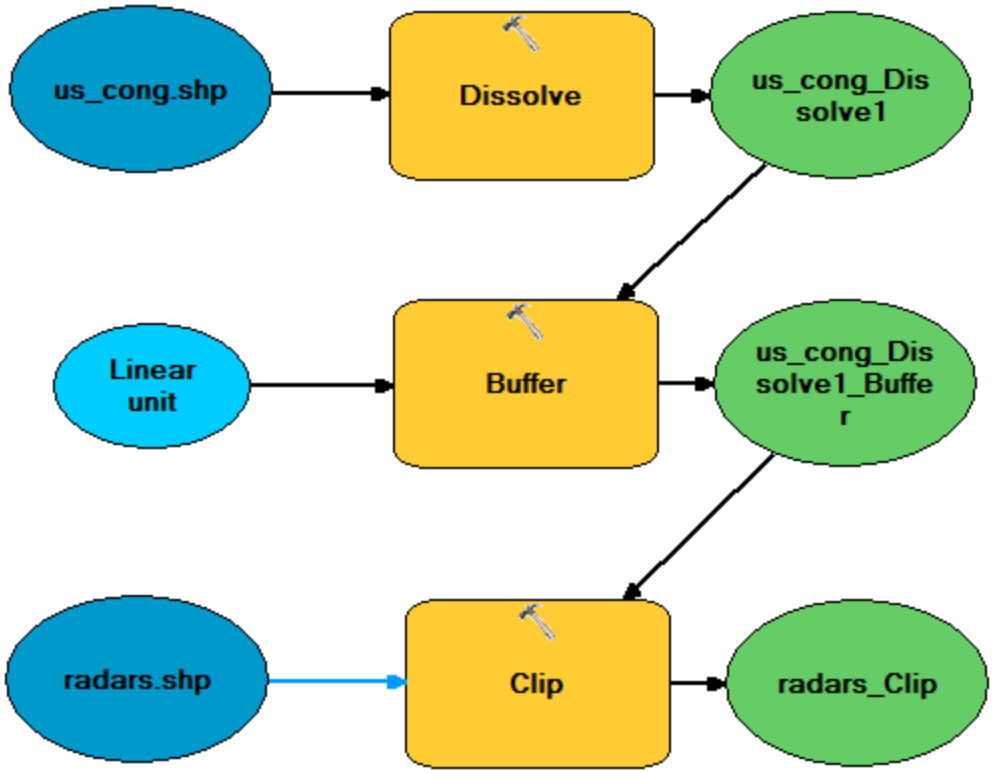


Adding an input value

- You could make the distance a model parameter or directly specify the value
  - Here, I specified 0.05 (units don't matter: it's map units)

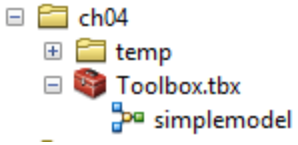


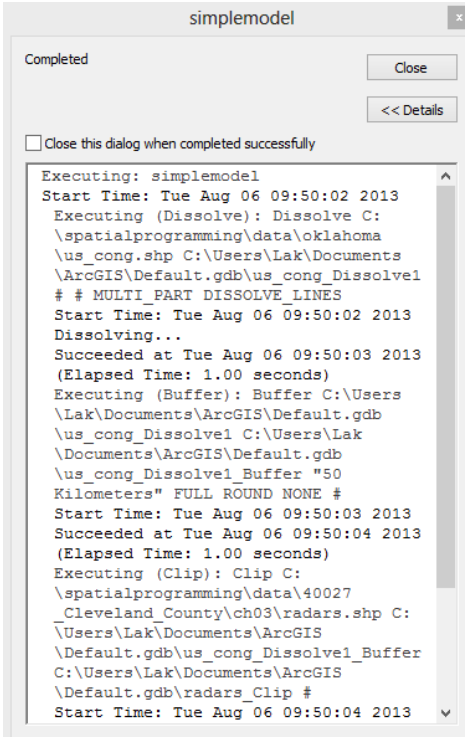
Final Model

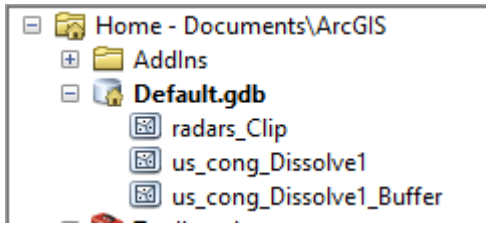

us_cong.shp
Dissolve
us_cong_Dissolve1
Linear unit
Buffer
us_cong_Dissolve1_Buffer
radars.shp
Clip
radars_Clip

Running the Model

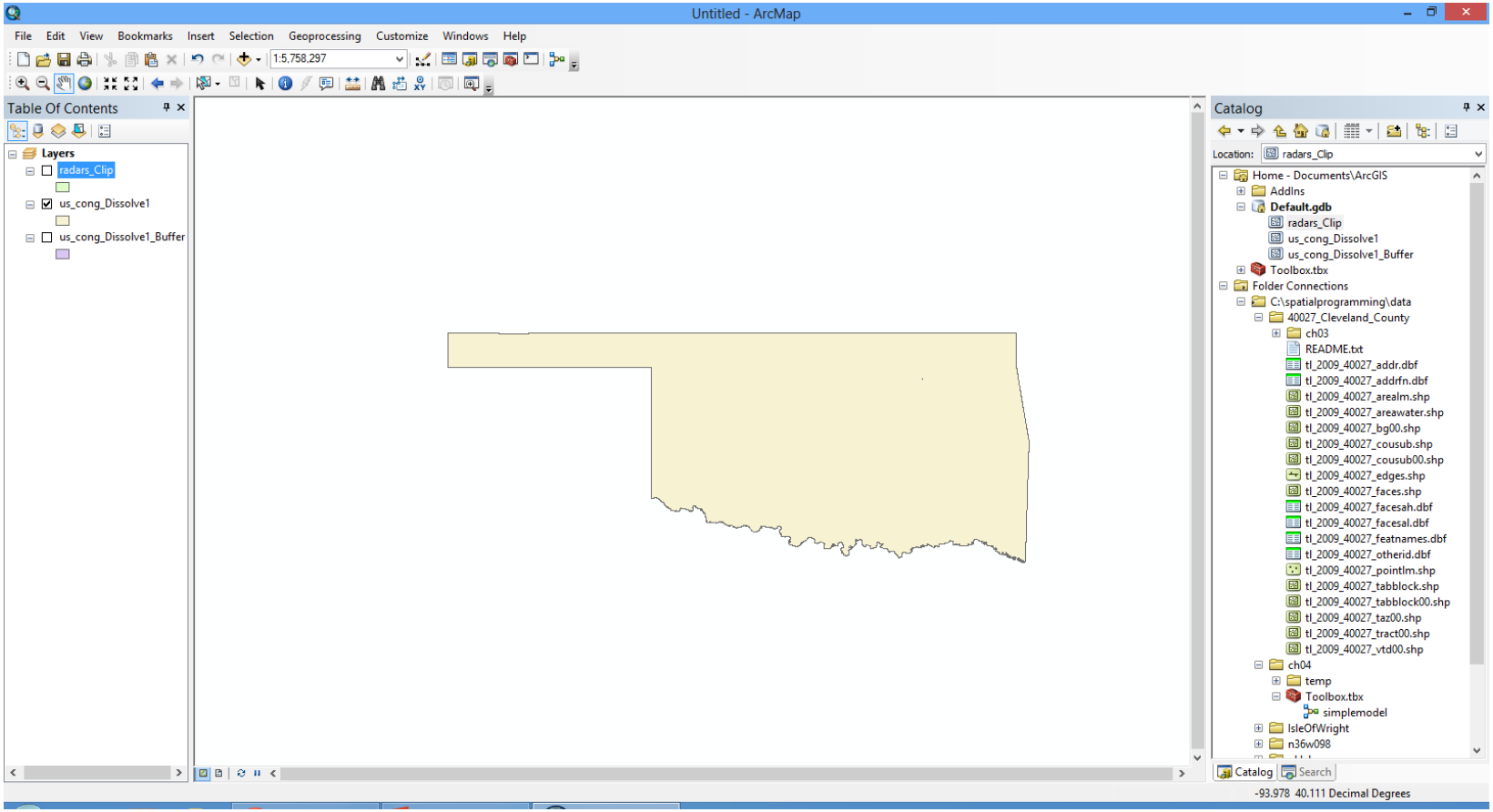
- Select Model | Run to run



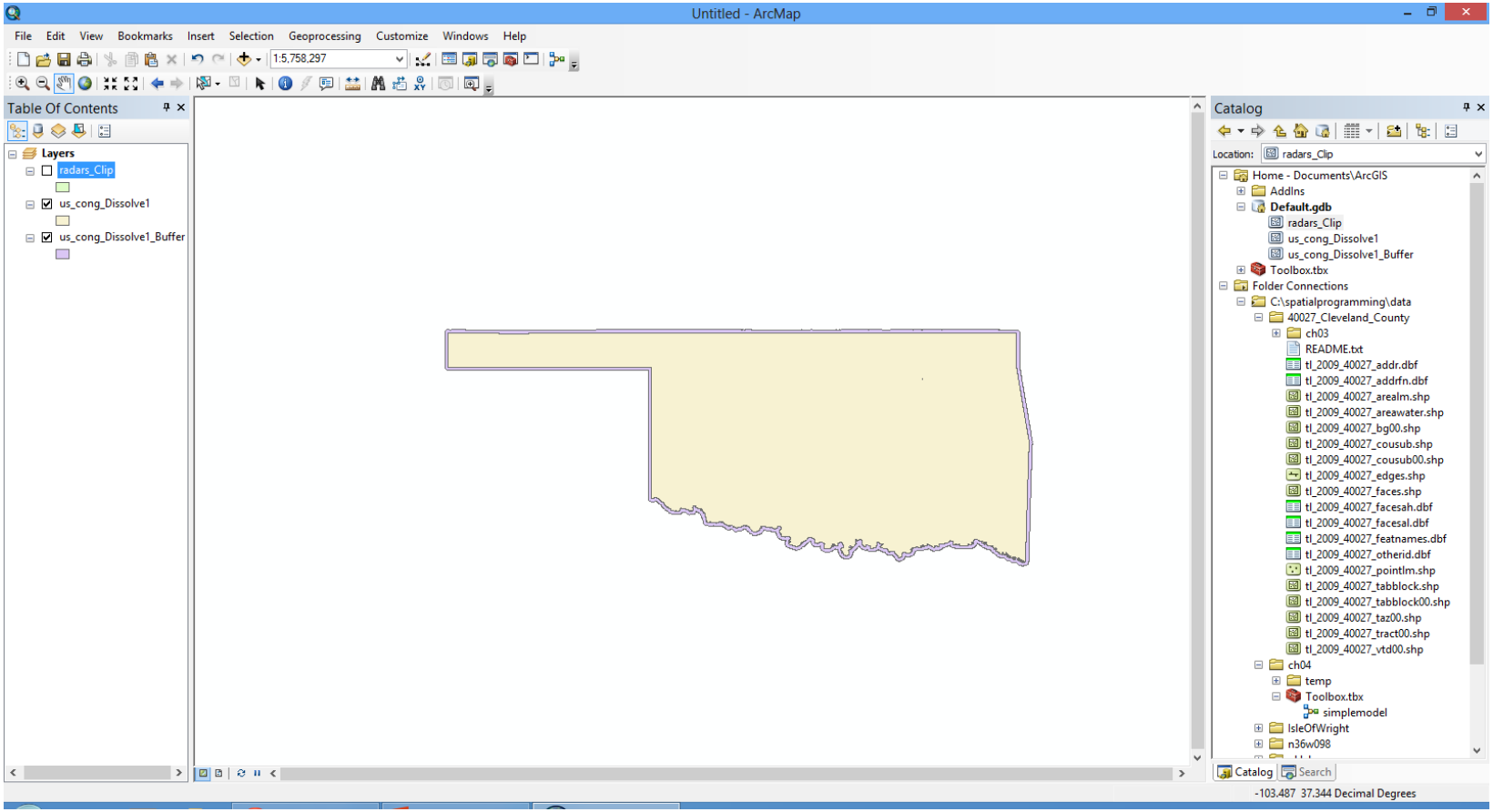




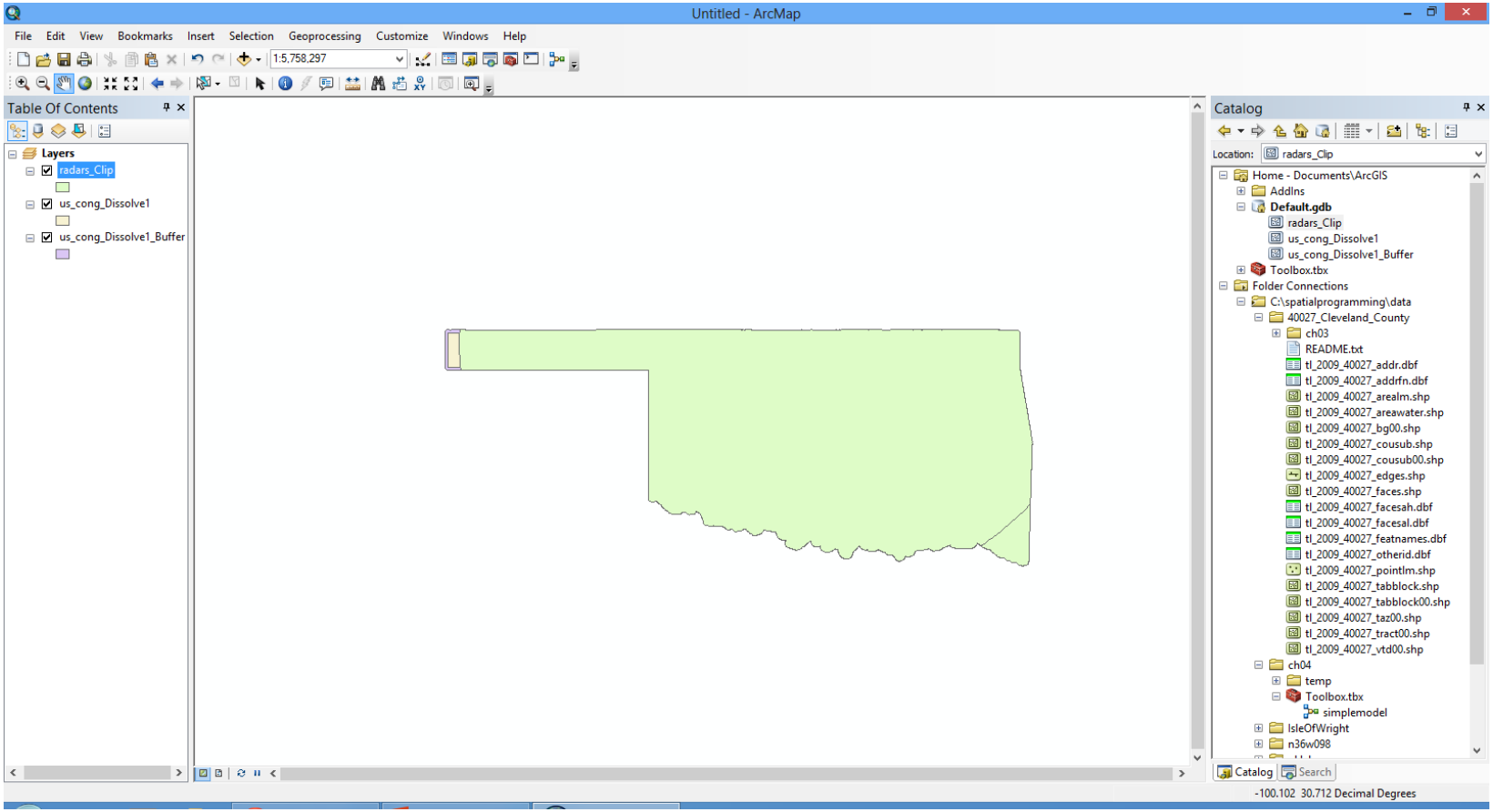
Dissolved


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:5,758,297
Table Of Contents
Layers
radars_Clip
us_cong_Dissolve1
us_cong_Dissolve1_Buffer
Catalog
Location: radars_Clip
Home - Documents\ArcGIS
AddIns
Default.gdb
radars_Clip
us_cong_Dissolve1
us_cong_Dissolve1_Buffer
Toolbox.tbx
Folder Connections
C:\spatialprogramming\data
40027_Cleveland_County
ch03
README.txt
tl_2009_40027_addr.dbf
tl_2009_40027_addrfn.dbf
tl_2009_40027_arealm.shp
tl_2009_40027_areawater.shp
tl_2009_40027_bg00.shp
tl_2009_40027_cousub.shp
tl_2009_40027_cousub00.shp
tl_2009_40027_edges.shp
tl_2009_40027_faces.shp
tl_2009_40027_facesah.dbf
tl_2009_40027_facesal.dbf
tl_2009_40027_featnames.dbf
tl_2009_40027_otherid.dbf
tl_2009_40027_pointlm.shp
tl_2009_40027_tabblock.shp
tl_2009_40027_tabblock00.shp
tl_2009_40027_taz00.shp
tl_2009_40027_tract00.shp
tl_2009_40027_vtd00.shp
ch04
temp
Toolbox.tbx
simplemodel
IsleOfWright
n36w098
Catalog Search
-93.978 40.111 Decimal Degrees

Buffered


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:5,758,297
Table Of Contents
Layers
radars_Clip
us_cong_Dissolve1
us_cong_Dissolve1_Buffer
Catalog
Location: radars_Clip
Home - Documents\ArcGIS
AddIns
Default.gdb
radars_Clip
us_cong_Dissolve1
us_cong_Dissolve1_Buffer
Toolbox.tbx
Folder Connections
C:\spatialprogramming\data
40027_Cleveland_County
ch03
README.txt
tl_2009_40027_addr.dbf
tl_2009_40027_addrfn.dbf
tl_2009_40027_arealm.shp
tl_2009_40027_areawater.shp
tl_2009_40027_bg00.shp
tl_2009_40027_cousub.shp
tl_2009_40027_cousub00.shp
tl_2009_40027_edges.shp
tl_2009_40027_faces.shp
tl_2009_40027_facesah.dbf
tl_2009_40027_facesal.dbf
tl_2009_40027_featnames.dbf
tl_2009_40027_otherid.dbf
tl_2009_40027_pointlm.shp
tl_2009_40027_tabblock.shp
tl_2009_40027_tabblock00.shp
tl_2009_40027_taz00.shp
tl_2009_40027_tract00.shp
tl_2009_40027_vtd00.shp
ch04
temp
Toolbox.tbx
simplemodel
IsleOfWright
n36w098
Catalog
Search
-103.487 37.344 Decimal Degrees

Clipped


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:5,758,297
Table Of Contents
Layers
radars_Clip
us_cong_Dissolve1
us_cong_Dissolve1_Buffer
Catalog
Location: radars_Clip
Home - Documents\ArcGIS
AddIns
Default.gdb
radars_Clip
us_cong_Dissolve1
us_cong_Dissolve1_Buffer
Toolbox.tbx
Folder Connections
C:\spatialprogramming\data
40027_Cleveland_County
ch03
README.txt
tl_2009_40027_addr.dbf
tl_2009_40027_addrfn.dbf
tl_2009_40027_arealm.shp
tl_2009_40027_areawater.shp
tl_2009_40027_bg00.shp
tl_2009_40027_cousub.shp
tl_2009_40027_cousub00.shp
tl_2009_40027_edges.shp
tl_2009_40027_faces.shp
tl_2009_40027_facesah.dbf
tl_2009_40027_facesal.dbf
tl_2009_40027_featnames.dbf
tl_2009_40027_otherid.dbf
tl_2009_40027_pointlm.shp
tl_2009_40027_tabblock.shp
tl_2009_40027_tabblock00.shp
tl_2009_40027_taz00.shp
tl_2009_40027_tract00.shp
tl_2009_40027_vtd00.shp
ch04
temp
Toolbox.tbx
simplemodel
IsleOfWright
n36w098
Catalog
Search
-100.102 30.712 Decimal Degrees

Python Scripts

Model to Script

- The simplest way to create a Python geoprocessing script in ArcGIS is to export a model
  - Right-click on a model in its toolbox and select Edit
  - In the Model graphic window that comes up
    - Model | Export | To Python Script
- All models can be converted to Python scripts
  - But not vice-versa (why?)

Running the Script

- Can simply double-click on the saved script to run it
  - Or open it in IDLE and run the module
    - Better since you can see error messages
- Try it now
  - Does it work?
  - What's wrong?

```
Traceback (most recent call last):
  File "C:\spatialprogramming\chapter04_arcpy\simplemodel.py
le>
    arcpy.Dissolve_management(us_cong_shp, us_cong_Dissolve1
T", "DISSOLVE_LINES")
  File "C:\Program Files (x86)\ArcGIS\Desktop10.1\arcpy\arcp
ne 3993, in Dissolve
    raise e
ExecuteError: Failed to execute. Parameters are not valid.
ERROR 000725: Output Feature Class: Dataset C:\Users\Lak\Doc
t.gdb\us_cong_Dissolve1 already exists.
Failed to execute (Dissolve).
```

The Script has hardcoded paths

simplemodel.py - C:\spatialprogramming\chapter04_arcpy\simplemodel.py

File Edit Format Run Options Windows Help

```
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------------
# simplemodel.py
# Created on: 2013-09-05 20:06:20.00000
#   (generated by ArcGIS/ModelBuilder)
# Description:
# ---------------------------------------------------------------------------

# Import arcpy module
import arcpy


# Local variables:
us_cong_shp = "C:\\spatialprogramming\\data\\oklahoma\\us_cong.shp"
Linear_unit = "0.05 Kilometers"
radars_shp = "C:\\spatialprogramming\\data\\40027_Cleveland_County\\ch03\\radars.shp"
us_cong_Dissolve1 = "C:\\Users\\Lak\\Documents\\ArcGIS\\Default.gdb\\us_cong_Dissolve1"
us_cong_Dissolve1_Buffer = "C:\\Users\\Lak\\Documents\\ArcGIS\\Default.gdb\\us_cong_Dissolve1_Buffer"
radars_Clip = "C:\\Users\\Lak\\Documents\\ArcGIS\\Default.gdb\\radars_Clip"

# Process: Dissolve
arcpy.Dissolve_management(us_cong_shp, us_cong_Dissolve1, "", "", "MULTI_PART", "DISSOLVE_LINES")

# Process: Buffer
arcpy.Buffer_analysis(us_cong_Dissolve1, us_cong_Dissolve1_Buffer, Linear_unit, "FULL", "ROUND", "NONE", "")

# Process: Clip
arcpy.Clip_analysis(radars_shp, us_cong_Dissolve1_Buffer, radars_Clip, "")
```

Changing Environment Variable

- The output file already exists from the time you ran Model
  - So, the script fails
- Modify the environment to allow the script to over-write:

```
# Import arcpy module
import arcpy

arcpy.env.overwriteOutput = True

# Local variables:
```

- Now run it again, within IDLE

Package

- Let's look at the generated script:
  - Which package does one need to import to use geoprocessing scripts in Python

```
import arcpy

us_cong_shp =
"C:\\spatialprogramming\\data\\oklahoma\\us_cong.shp"
us_cong_Dissolve1 =
"C:\\Users\\Lak\\Documents\\ArcGIS\\Default.gdb\\us_cong_Dis
solve1"
arcpy.Dissolve_management(us_cong_shp, us_cong_Dissolve1,
"", "", "MULTI_PART", "DISSOLVE_LINES")
```

- Older scripts may import the package ArcGISscripting

Adding output messages

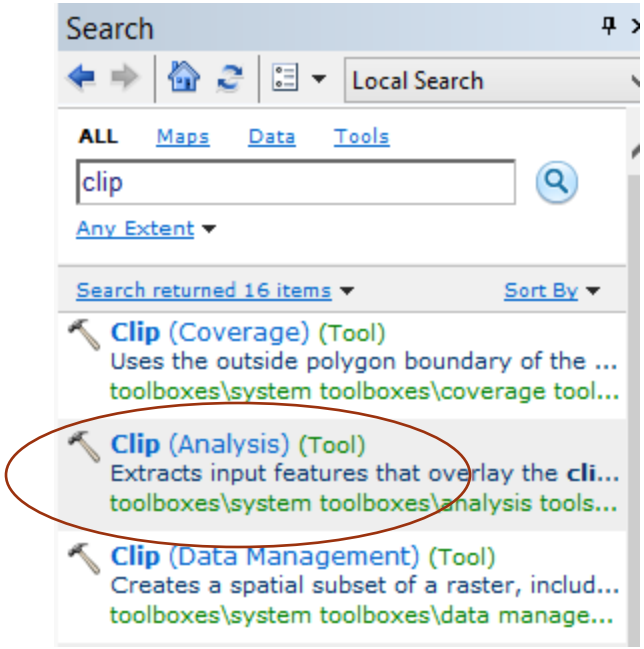
- The script ran silently
  - Better to have it provide status as it goes along

```
arcpy.Dissolve_management(us_cong_shp, us_cong_Dissolve1,
"", "", "MULTI_PART", "DISSOLVE_LINES")

print arcpy.GetMessages()
```

- Try it again

Function names

- The name of a function corresponds to how it is listed in the Catalog Search window:




```
arcpy.Clip_analysis(…)
```

This alternative method also works:

```
arcpy.analysis.Clip(…)
```

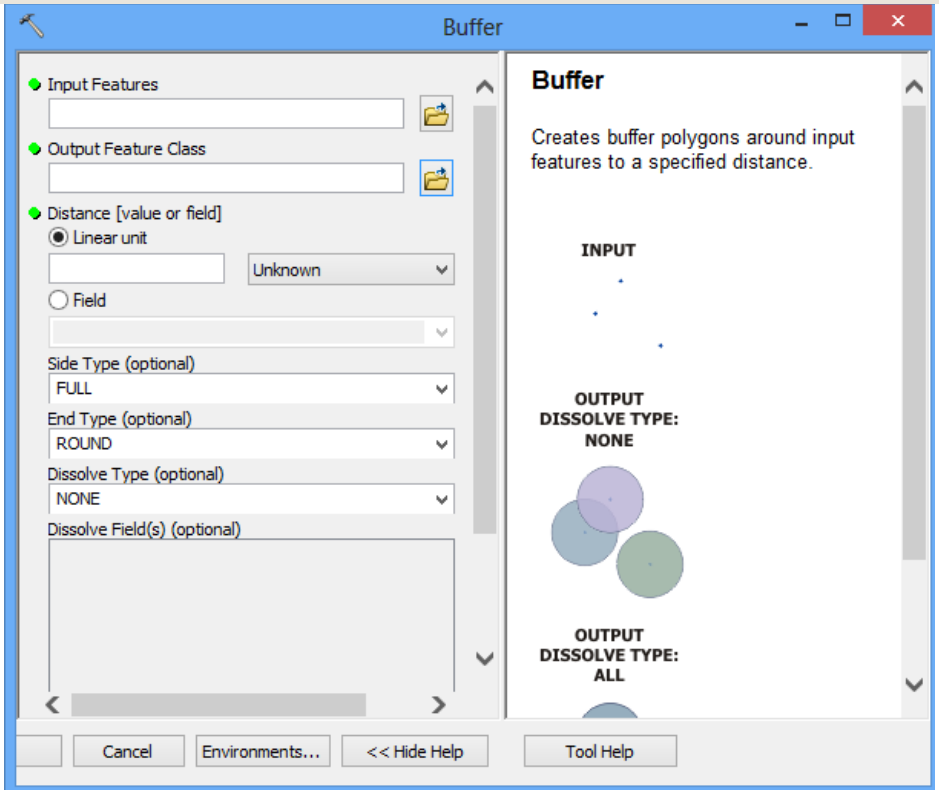
Data Management is simply "management":

```
arcpy.Dissolve_management
```

Function parameters

- Function parameters are in the same order as in the dialog:

```
arcpy.Buffer_analysis(us_cong_Dissolve1,
us_cong_Dissolve1_Buffer, Linear_unit, "FULL", "ROUND",
"NONE", "")
```



Optional parameters

- Can omit all the optional parameters:

```
arcpy.Buffer_analysis(us_cong_Dissolve1,
us_cong_Dissolve1_Buffer, Linear_unit)
```

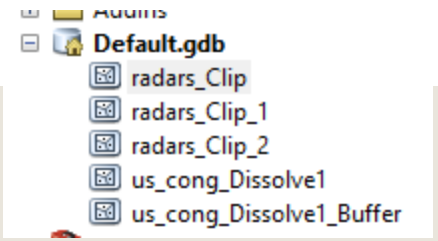
- Or specify only some of them:

```
arcpy.Buffer_analysis(us_cong_Dissolve1,
us_cong_Dissolve1_Buffer, Linear_unit, "", "ROUND")
```

  - Can use either an empty string (as above) or the special character (“#”) to denote the default value for a parameter

Why a script?

- At this point, we have not done anything in the Python script that the model didn't already do
- Now, let's add some looping and smart processing
  - Run the script




```
for i in range(1,3):
    bufferBy = i * 0.05
    Linear_unit = str(bufferBy) + " Kilometers"
    # Process: Buffer
    arcpy.Buffer_analysis(us_cong_Dissolve1, us_cong_Dissolve1_Buffer,
Linear_unit, "FULL", "ROUND", "NONE", "")
    print arcpy.GetMessages()

    # Process: Clip
    out = radars_Clip + "_" + str(i)
    arcpy.analysis.Clip(radars_shp, us_cong_Dissolve1_Buffer, out, "")
    print arcpy.GetMessages()
```

Odds and Ends

Non-standard tools

- The package arcpy imports all the system toolboxes
  - What if you have a custom toolbox?

```
import arcpy

arcpy.ImportToolbox( "C:/Data/sometoolbox.tbx",  "sometool" )
```

Actual location of the toolbox on the file system (can also use relative path or Internet URL)

The alias you are going to use for this toolbox

  - To use a function ComputeDistance in this toolbox:

```
arcpy.sometool.ComputeDistance( … )

print help( arcpy.sometool.ComputeDistance )
```

Tools vs. Non-tools

- ArcPy distinguishes between tools and non-tools

Tools	Non-tools
Part of a toolbox	Just a function
Return a Result object	Do not return a Result (can return something else)
Produce geoprocessing log messages	No messages

```
if ( arcpy.Exists("C:/data/somefile.shp") ):
    # etc.

desc = arcpy.Describe( "C:/data/somefile.shp" )
print desc.datasetType
```

Creating objects

- Some functions require objects instead of files
  - To create objects:

```
sref = arcpy.SpatialReference( "c:/data/somefile.prj" )

arcpy.management.CreateFeatureClass(
       "C:/data", #outpath
       'out.shp',
       'POLYLINE',
       "","","",
       sref )
```

  - Done this way because the spatial reference is itself just a long prj string and we might want to pass the actual value rather than the filename

Licenses

- When calling a function, arcpy checks for license

```
arcpy.sa.Slope( "c:/data/demo", "DEGREE" )
```

  - Will generate an error if the Spatial Analyst extension is not licensed by the user
- Can check for, and use, a product or extension:

```
if arcpy.CheckExtension( "spatial" ) == "Available":
    arcpy.CheckOutExtension( "spatial" )
    arcpy.sa.Slope …
    arcpy.CheckInExtension( "spatial" )
```

License for products

- Can similarly check for product licenses:

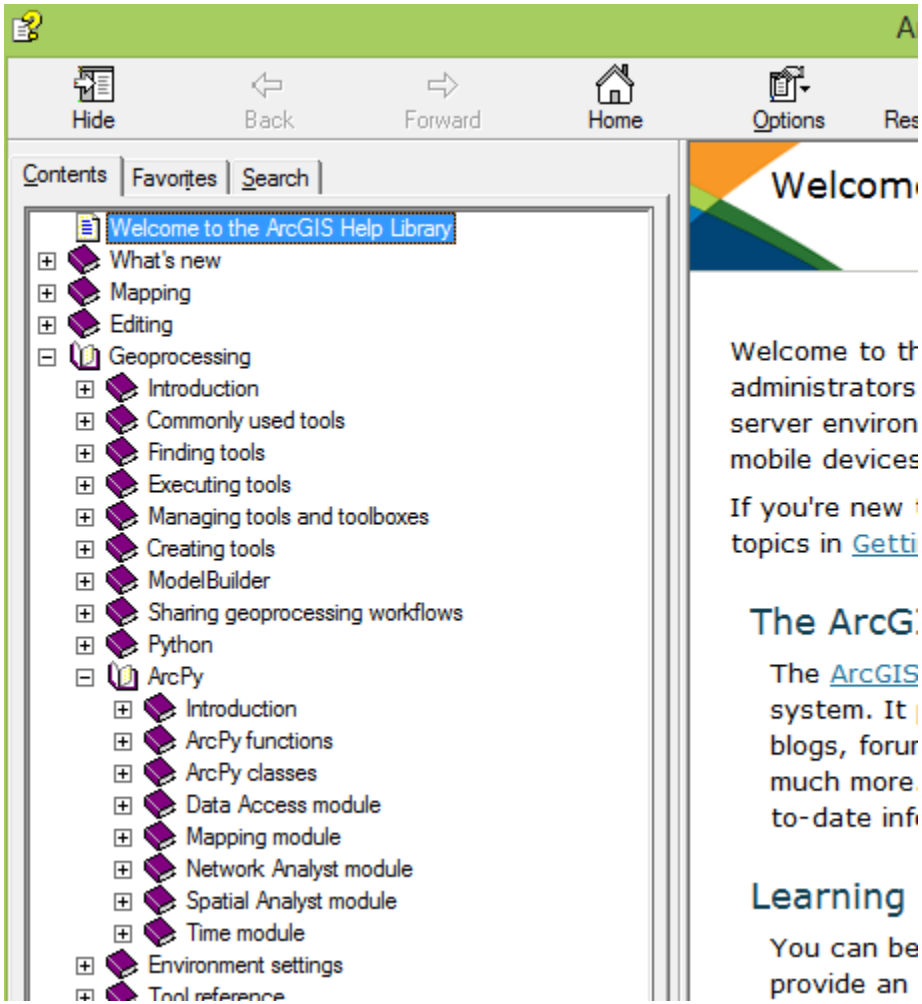
```
if arcpy.CheckProduct( "arcinfo" ) == "Available":
    arcpy.CheckOutProduct("arcinfo")
    # do stuff with arcinfo

    arcpy.CheckInProduct("arcinfo")
```

- But importing a module automatically checks out that module, so could as well do:

```
import arcpy
import arcinfo
```

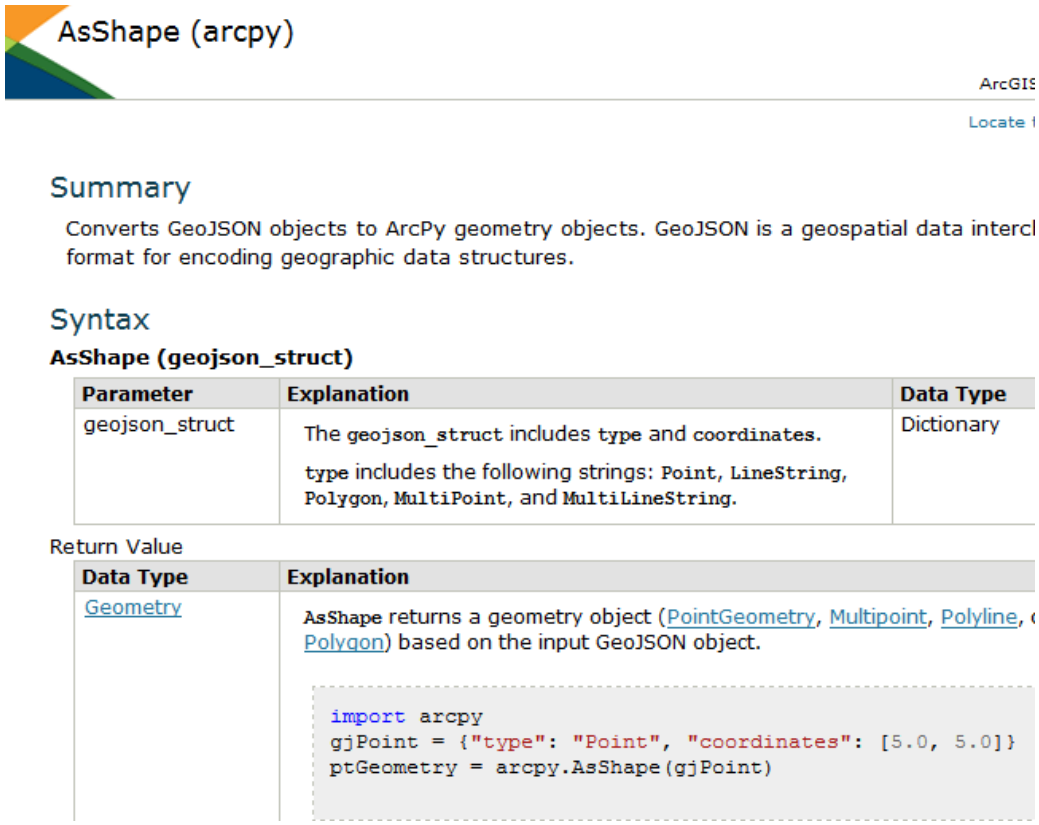
  - Other products include arcserver, arceditor, arcview, engine, enginegeodb

Help (Help | ArcGIS Desktop Help)



Help on a function

- Includes sample code
  - ArcPy |
  - ArcPy Functions |
  - Geometry |
  - AsShape
- Try out code to create a polygon shape



AsShape (arcpy)

ArcGIS

Locate

Summary

Converts GeoJSON objects to ArcPy geometry objects. GeoJSON is a geospatial data intercl format for encoding geographic data structures.

Syntax

AsShape (geojson_struct)

Parameter	Explanation	Data Type
geojson_struct	The `geojson_struct` includes `type` and `coordinates`. `type` includes the following strings: `Point`, `LineString`, `Polygon`, `MultiPoint`, and `MultiLineString`.	Dictionary

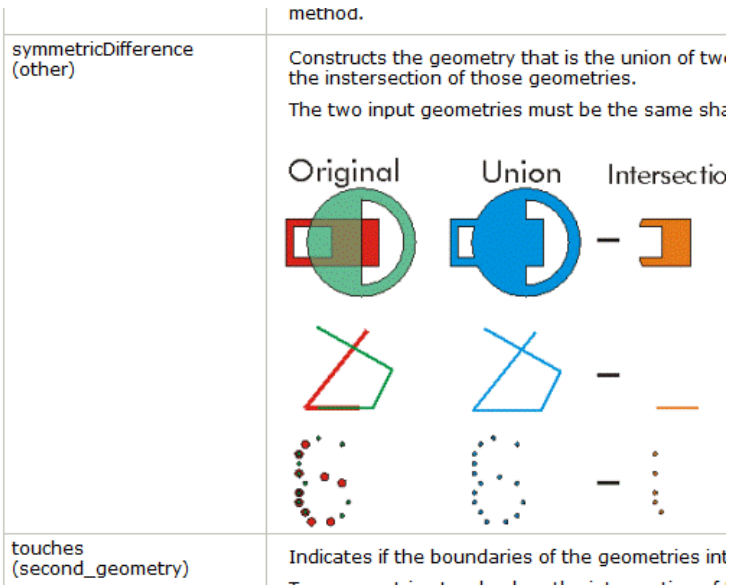
Return Value

Data Type	Explanation
Geometry	**AsShape** returns a geometry object (PointGeometry, Multipoint, Polyline, Polygon) based on the input GeoJSON object.

```
import arcpy
gjPoint = {"type": "Point", "coordinates": [5.0, 5.0]}
ptGeometry = arcpy.AsShape(gjPoint)
```

Help on a class

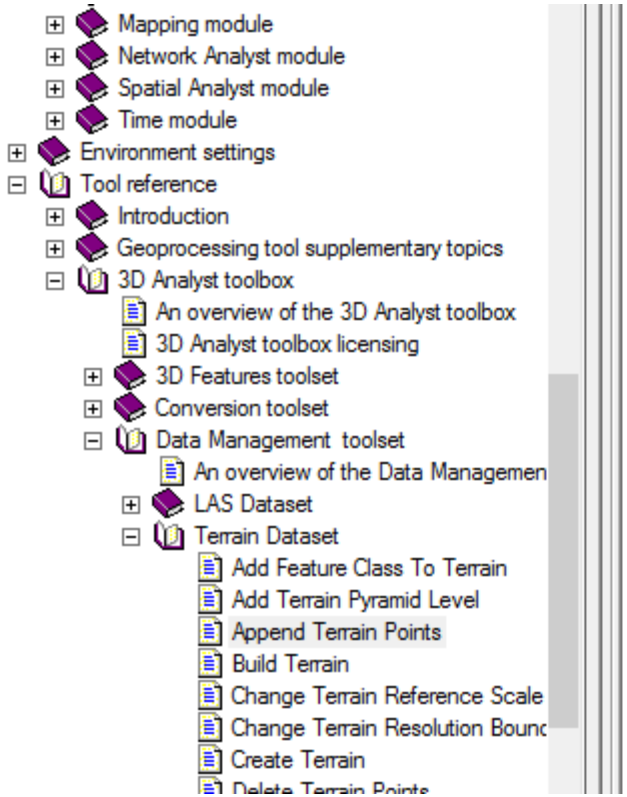
- Methods are explained in detail
  - ArcPy |
  - ArcPy classes |
  - Geometry |
  - Polygon
- Try out code to check if one polygon overlaps with another



Polygon …

```
import arcpy

poly1 = {
    "type" : "Polygon",
    "coordinates": [
            [[35.7,-98.1], [35.8,-98.1], [35.8, -99.1], [35.7, -99.1], [35.7,-98.1]]
        ]
    }

poly2 = {
    "type" : "Polygon",
    "coordinates": [
            [[35,-98], [38,-98], [38, -99], [35, -99], [35,-98]]
        ]
    }

shape1 = arcpy.AsShape(poly1)
shape2 = arcpy.AsShape(poly2)

if shape1.overlaps(shape2):
    print "The two polygons overlap"
else:
    print "The two polygons do not overlap"
```

Help on a tool



results in all the points from the input feature class being loaded to the terrain feature.

Code Sample

AppendTerrainPoints example 1 (Python window)

The following sample demonstrates the use of this tool in the Python window:

```
import arcpy
from arcpy import env

arcpy.CheckOutExtension('3D')
env.workspace = 'C:/data'
arcpy.AppendTerrainPoints_3d('sample.gdb/featuredataset/terrain',
                             'existing_points', 'new_points.shp')
```

AppendTerrainPoints example 2 (stand-alone script)

The following sample demonstrates the use of this tool in a stand-alone Python script

Homework

- Using ModelBuilder, build a Model to accomplish some task (your choice, but can not be a simple repeat or subset of my dissolve-buffer-clip example)
- Convert it into a Python script
- Modify the script to do something not easily achievable with just ModelBuilder
- Submit a PDF containing Model Builder graphic, python script, and snapshots of inputs and outputs

Data files

Listing

- While we can use Python file and directory handling, using arcpy is more convenient because it knows what extensions are what type

```
arcpy.env.workspace = "c:/data"
features = arcpy.ListFeatureClasses( "usa_*" )

for feature in features:
    # etc.
```

  - ListFeatureClasses(), ListRasters(), ListIndexes(), ListDatasets(), etc.

```
rasters = arcpy.ListRasters( "usa_*" , "tif")
```

    - tif because that is the file extension

Listing Fields in a Shapefile

- Fields in a shapefile:

```
import arcpy
arcpy.env.overwriteOutput = True
arcpy.env.workspace = "C:\\spatialprogramming\\data\\40027_Cleveland_County"

watershp = "tl_2009_40027_areawater.shp"

if not arcpy.Exists(watershp):
    print watershp + " does not exist!"
    exit

fields = arcpy.ListFields(watershp, "", "")
for field in fields:
    print "{:s} is {:d} characters".format(field.name, field.length)
```

```
FID is 4 characters
Shape is 0 characters
STATEFP is 2 characters
COUNTYFP is 3 characters
ANSICODE is 8 characters
HYDROID is 22 characters
FULLNAME is 100 characters
MTFCC is 5 characters
ALAND is 14 characters
AWATER is 14 characters
INTPTLAT is 11 characters
INTPTLON is 12 characters
```

Working with tables

- To work with .dbf data, use cursors
  - Search cursor to select (retrieve) rows
  - Insert cursor to insert new rows
  - Update cursor to update/delete rows

Search cursor

- Note the "with" syntax to prevent resource leaks:

```
with arcpy.da.SearchCursor(watershp, ["FID","FULLNAME"]) as cursor:
    for row in cursor:
        fid = row[0]
        fullname = row[1]
        if len(fullname.strip()) > 0:
            print "{:d} is {:s}".format(fid, fullname)
```

```
2619 is Sleepy Hollow Lk
2620 is Robinson Bay
2625 is Canadian Riv
2646 is Canadian Riv
3216 is Dahlgren Lk
```

Update

- Will modify all the necessary files:

```
with arcpy.da.UpdateCursor(watershp, ["FID","FULLNAME"]) as cursor:
    for row in cursor:
        fid = row[0]
        fullname = row[1]
        if len(fullname.strip()) == 0:
            cursor.deleteRow()
```

Insert

- Insert will append NULL strings for all other values
  - When inserting, make sure the name doesn't have invalid characters in it (Invalid chars are replaced by an _)

```
cursor = arcpy.data.InsertCursor(watershp, ["FULLNAME"])
name = arcpy.ValidateFieldName("Lake Lak?")
cursor.insertRow(name)
```

- When creating files, however, can ask arcpy to create a unique name (formed by adding 0/1/2/etc. to name)

```
out = arcpy.CreateUniqueName( "clipped.shp" )
```

Geometry files

Shape tokens

- Can use the cursor to get at shapes
  - Special tokens Length and Area are available (in Map units)

```
with arcpy.da.SearchCursor(watershp, ["SHAPE@LENGTH", "SHAPE@AREA", "FULLNAME", "AWATER"]) as cursor:
    for row in cursor:
        length = row[0]
        area = row[1]
        fullname = row[2]
        areawater = row[3]
        if len(fullname.strip()) > 0:
            print "{:s}  length={:.6f} area={:.6f} awater={:.1f}".format(fullname, length, area, areawater)
```

```
Canadian Riv  length=0.002333 area=0.000000 awater=3112.0
Mussel Shoals Lk  length=0.022148 area=0.000008 awater=81903.0
Canadian Riv  length=0.706529 area=0.000183 awater=1846584.0
Blue Lk  length=0.007219 area=0.000002 awater=17532.0
Odon Lk  length=0.022217 area=0.000010 awater=99846.0
Bishop Lk  length=0.009151 area=0.000002 awater=18072.0
Kitchen Lk  length=0.018271 area=0.000010 awater=99841.0
Sleepy Hollow Lk  length=0.006470 area=0.000002 awater=15838.0
Robinson Bay  length=0.007506 area=0.000002 awater=23176.0
Canadian Riv  length=0.005282 area=0.000000 awater=3908.0
Canadian Riv  length=0.017335 area=0.000010 awater=102074.0
```

- SHAPE@XY
  - Available for Point files

Creating a shapefile