Exercise 5: Working with Geoprocessor Tools

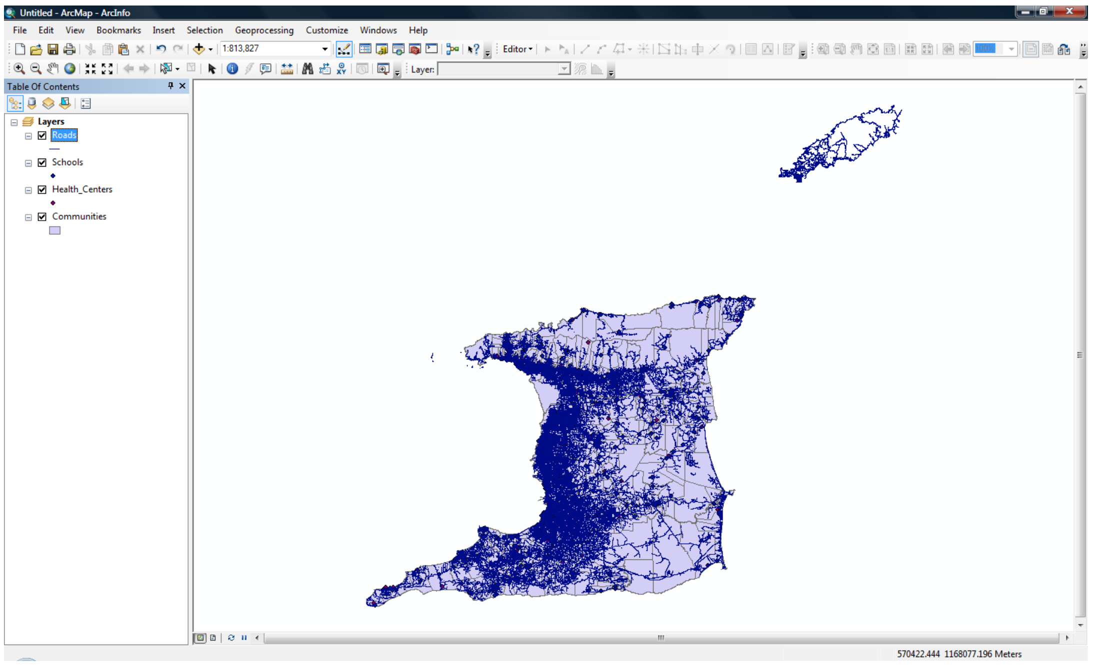
No matter what license level of ArcGIS Desktop (ArcView, ArcEditor, ArcInfo) you happen to be using you will have a number of geoprocessing tools at your disposal to accomplish many GIS tasks. Although the number of tools available at each licensing level varies, ArcGIS 10 provides a common geoprocessing framework for ArcView, ArcEditor, and ArcInfo. ArcView supports a core set of simple data loading and translation tools as well as approximately 40 fundamental analysis tools. ArcEditor augments this set with tools for geodatabase creation and loading. ArcInfo supplies approximately 800 tools including all geoprocessing functionality that has been available in ArcInfo Workstation. ArcInfo supports advanced analysis and complex work flows. Additional geoprocessing tools are supplied by ArcGIS extensions. For example, ArcGIS Spatial Analyst and ArcGIS 3D Analyst add more than 200 tools such as raster modeling tools. In this exercise we will explore ArcToolbox which is the ArcGIS container for geoprocessing tools and see how you can call these tools from your Python geoprocessing scripts.

In this exercise you are going to learn how to work with these tools in your Python scripts.
At the end of this exercise you will have learned the following:

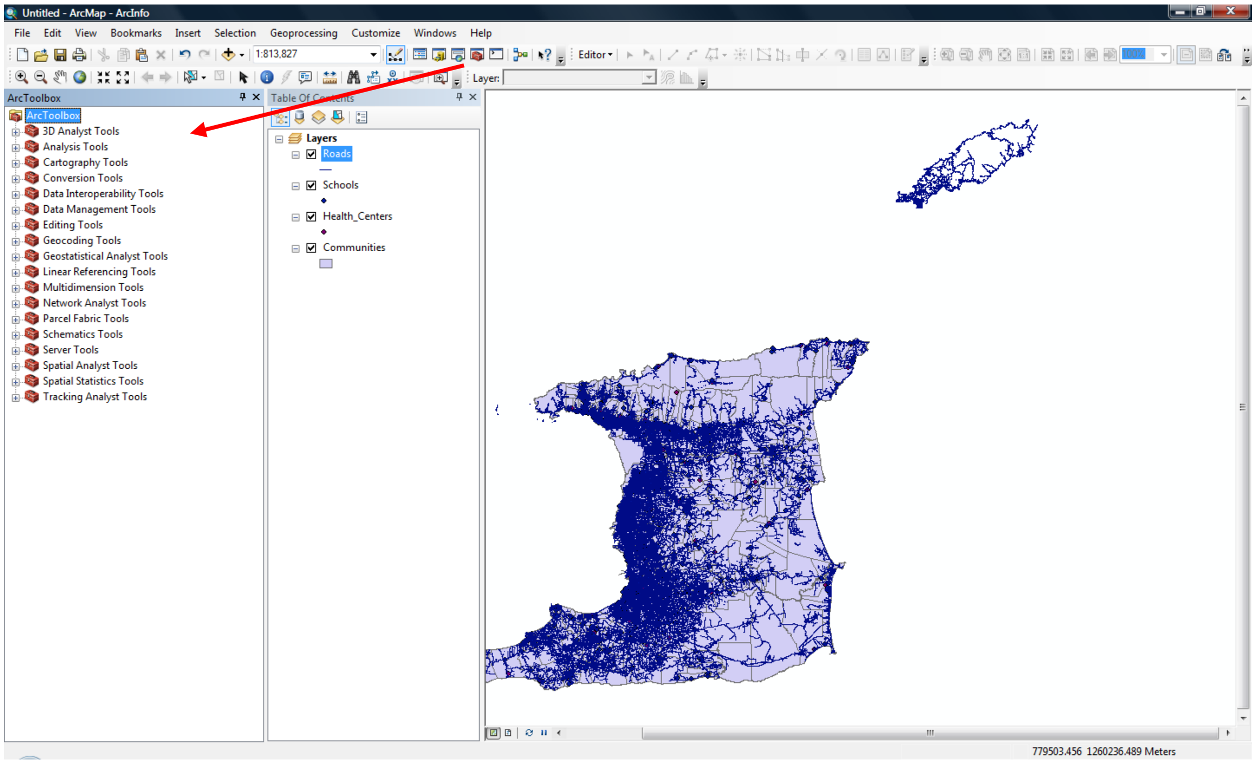
- How to examine environment settings
- Find available geoprocessing tools
- Understand how to call tools in your scripts by referencing the toolbox name
- Use tool help for syntax information
- Load pre-written python code from your hard disk
- Run tools from your scripts

Step 1: Adding data

- Connect to the Exercise4 personal geodatabase and add the following datasets; roads, schools, health centers and communities.

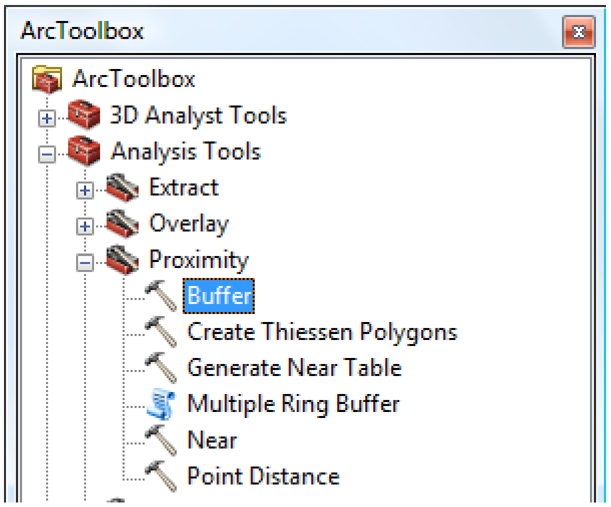

Untitled - ArcMap - ArcInfo
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:813,827
Editor
Layer:
Table Of Contents
Layers
Roads
Schools
Health_Centers
Communities
570422.444 1168077.196 Meters

Step 2: Open the ArcToolbox window using the show/hide ArcToolbox window

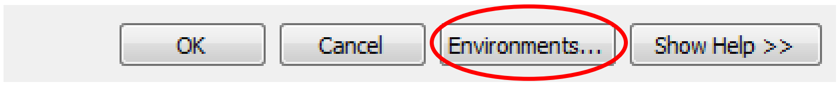


Step 3: Examine Environment Settings
There are many temporary environment settings that you can set. Environment settings can be thought of as additional parameters that a tool can use to affect its results. They differ from normal parameters in that they don't appear on a tool's dialog. Rather, they are values you set once using a separate dialog (or inside your script), and are then used by tools when they are run. Let's take a look at an example tool and environment settings.

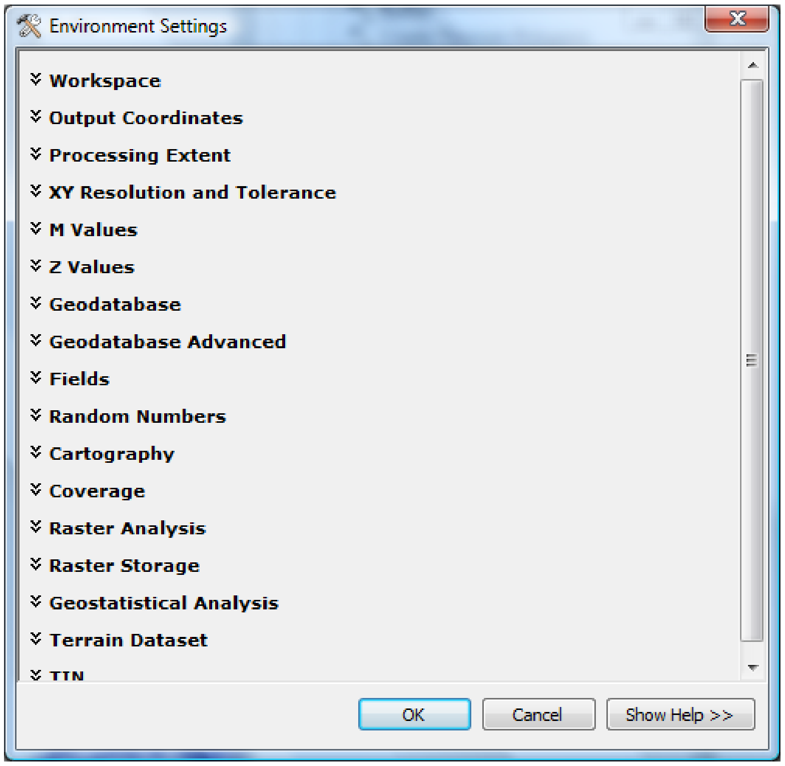
- In the ArcToolbox window select Analysis Tool →Proximity. You should see several tools including Buffer, Near, Point Distance, and perhaps several others. We'll take a look at the Buffer tool.



- Double-click Buffer. This is an example of a tool that can be run from your Python scripts or by manually supplying the requested information. Notice the Environments button at the bottom of the dialog.



- Click the Environments button. The Environment Settings dialog will display. Notice that the environment settings are grouped according to function.



- Take some time to explore the environment settings that you can use.

Step 4: Explore Available Tools
Spend some time examining the tools you have available. Remember that the tools at your disposal will be dependent upon your ArcGIS Desktop license level. Tools are grouped categorically by functional type into Toolboxes and Toolsets. For instance, the Analysis Tools toolbox contains toolsets for Extract, Overlay, Proximity, and Statistics. Some of the more commonly used toolboxes include Analysis Tools, Conversion Tools, Geocoding Tools, and Data Management Tools. Individual tools can be found in these toolboxes.

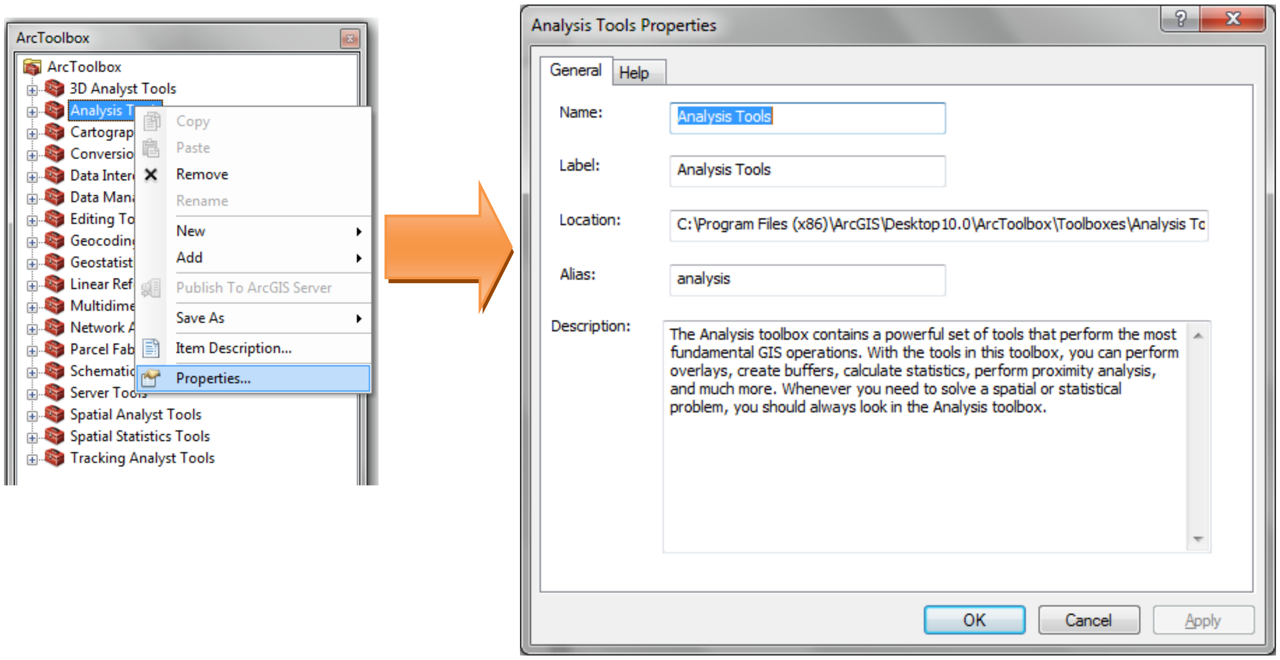
- What tool in the Data Management Tools toolbox would you use to create a new feature class? ____

- In which toolset is this tool located?
  ____

- What functional grouping of environment settings can be used for this tool?
  ____

Step 5: Toolbox Names and Aliases
Each toolbox in ArcToolbox has a name and an alias. In this step we'll examine these concepts.

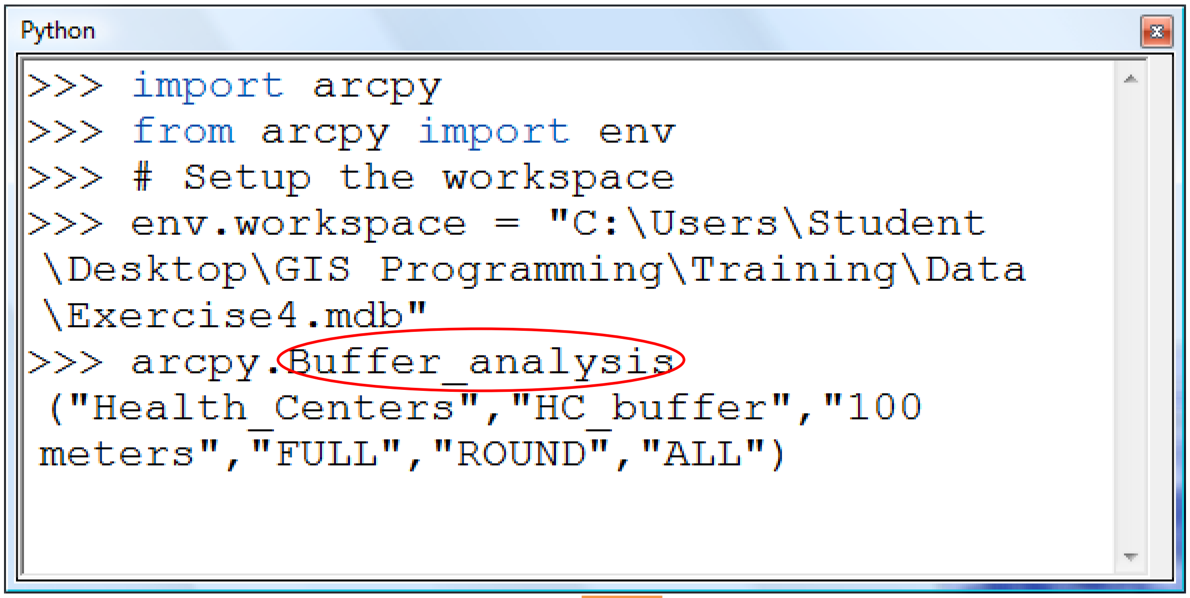
- In ArcToolbox right click the Analysis Tools toolbox and select Properties. Notice that the alias for the Analysis Tools toolbox is 'analysis'.



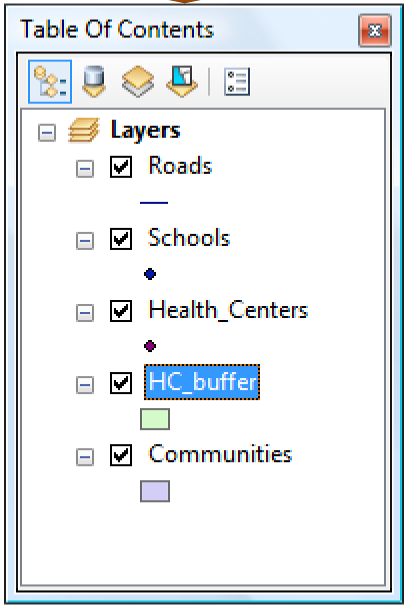
Name: Analysis Tools
Label: Analysis Tools
Location: C:\Program Files (x86)\ArcGIS\Desktop10.0\ArcToolbox\Toolboxes\Analysis Tc
Alias: analysis
Description: The Analysis toolbox contains a powerful set of tools that perform the most fundamental GIS operations. With the tools in this toolbox, you can perform overlays, create buffers, calculate statistics, perform proximity analysis, and much more. Whenever you need to solve a spatial or statistical problem, you should always look in the Analysis toolbox.

You must add the alias of the toolbox to the end of the tool name when calling a tool from your script. This is due to duplicate naming of tools in the Geoprocessing framework. So, make sure that you add an underscore plus the toolbox alias when calling a geoprocessing tool as seen in the code example below. The name of the tool to be run is specified first followed by an underscore and then the toolbox alias followed by the parameters.

- Create a 100 meter buffer around all health centers. Use the syntax in the figure below to guide you.



```
>>> import arcpy
>>> from arcpy import env
>>> # Setup the workspace
>>> env.workspace = "C:\Users\Student
 \Desktop\GIS Programming\Training\Data
 \Exercise4.mdb"
>>> arcpy.Buffer_analysis
 ("Health_Centers","HC_buffer","100
meters","FULL","ROUND","ALL")
```



The new layer with the given Output Feature Class name should now appear in the Table of Contents sections of ArcMap. You may need to zoom into the layer near health centers to see the creates buffers.

Step 6: Tool Syntax
Each tool in ArcToolbox has a specific syntax that you'll need to use when calling the tool to perform a specific function. There are a number of help resources that you can use to obtain syntax help for calling a tool. These include ArcGIS Desktop help, ArcToolbox help, and Python syntax help. For this exercise we'll use the ArcGIS Desktop help system, but realize that the syntax definition will be the same no matter what help system you use.

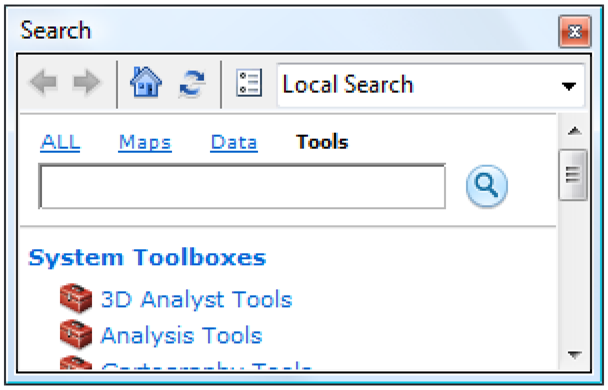
- In the Analysis Tools toolbox, go to the Overlay toolset and right click the Intersect tool. Select Help. This will open the ArcGIS Desktop help system to the tool you have selected. Each tool will contain an illustration of the function provided by the tool, usage tips, command line syntax, scripting syntax and code examples.

- What is the scripting syntax for the Intersect tool?

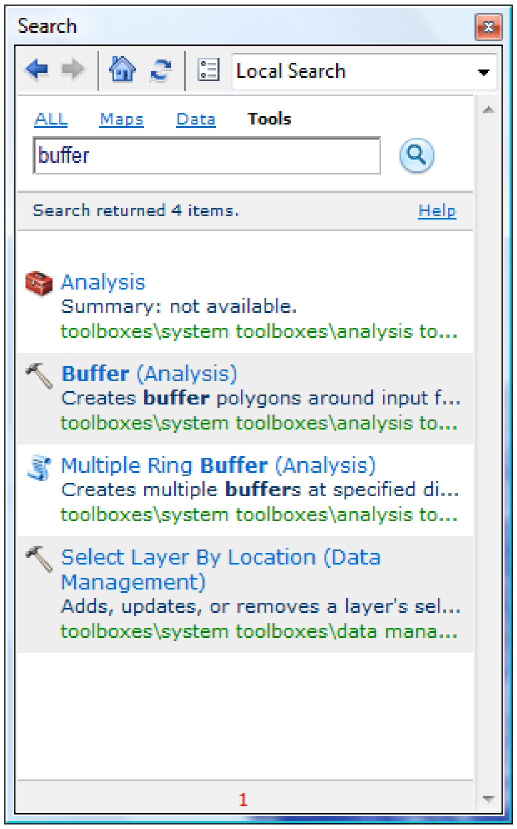
  ____

Step 7: Searching for a Tool
With so many tools available in ArcToolbox it is sometimes difficult to find the tool you're looking for. ArcGIS Desktop 10 has a nice tool that helps you search for the tool you need.

- In ArcMap select the Geoprocessing menu item and then Search For Tools. This will display the Search Window as seen below.



- Type 'Buffer' in the Search box and hit the search button.



buffer

Search returned 4 items.

Analysis
Summary: not available.
toolboxes\system toolboxes\analysis to...

Buffer (Analysis)
Creates **buffer** polygons around input f...
toolboxes\system toolboxes\analysis to...

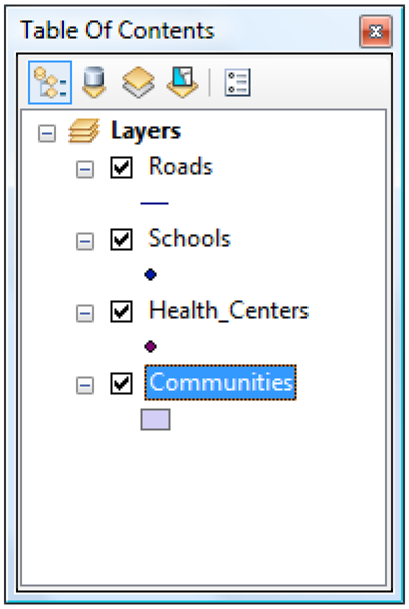
Multiple Ring **Buffer** (Analysis)
Creates multiple **buffers** at specified di...
toolboxes\system toolboxes\analysis to...

Select Layer By Location (Data Management)
Adds, updates, or removes a layer's sel...
toolboxes\system toolboxes\data mana...

- The search should return several items including the Buffer tool located in the Analysis toolbox. As you mouse over each link the help system will return additional information about the tool. You can click the link to display the user interface dialog for the tool. Right now we're not interested in running the tool from the user interface though. I just wanted to show you how to search for tools should you need more information about what they do or where they've found within ArcToolbox. What we're interested in doing in this class is actually running these tools from our Python scripts.

Step 8: Example

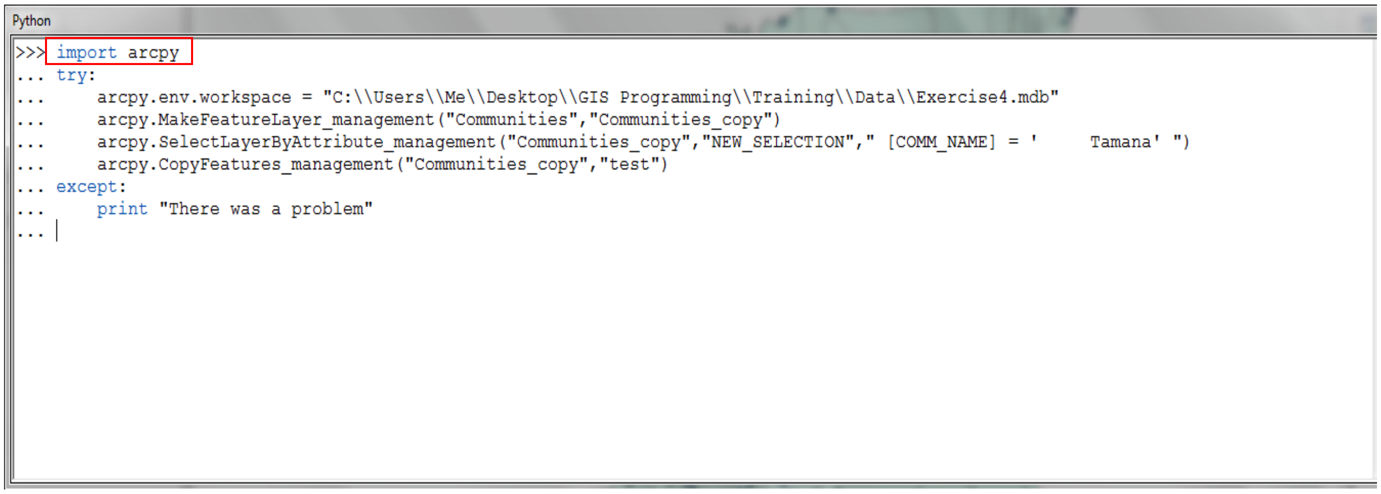
Now that you understand the fundamental tool concepts in ArcToolbox we will apply this knowledge to the Python scripting environment by calling various tools from a script. What we're going to do is create a new polygon feature layer (shapefile) containing a subset of communities from an existing layer in your project called "Communites".



Specifically, we are going to query the "Communities" layer for all communities with the community name Tamana. These features will then be copied to a new feature layer. We are going to use three tools to accomplish this task, all of which are found in the Data Management toolbox.

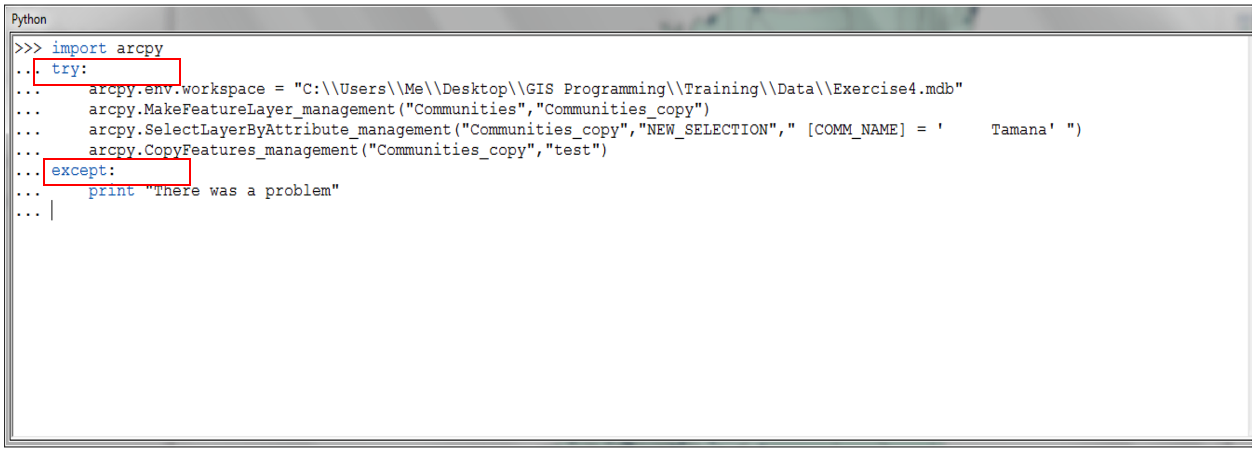
- Open notepad (or some other preferred text editor). Type the lines of code which follow as you read through this exercise.

- Import the ArcPy module and create a new instance of the arcgisscripting object.



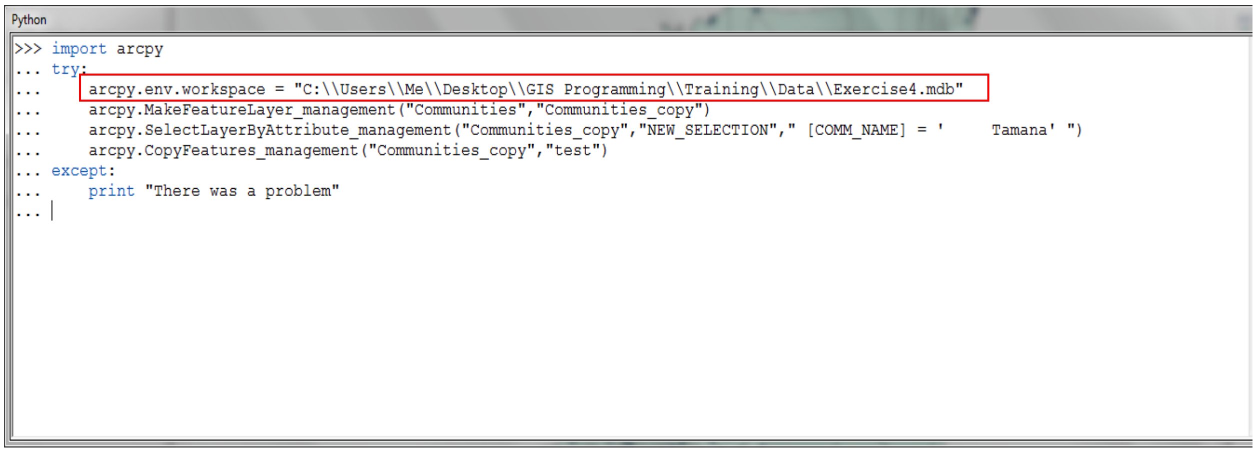
```
>>> import arcpy
... try:
...     arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data\\Exercise4.mdb"
...     arcpy.MakeFeatureLayer_management("Communities","Communities_copy")
...     arcpy.SelectLayerByAttribute_management("Communities_copy","NEW_SELECTION"," [COMM_NAME] = '     Tamana' ")
...     arcpy.CopyFeatures_management("Communities_copy","test")
... except:
...     print "There was a problem"
... 
```

In Python you can use try/except statements to wrap your code into a structure that allows you to capture any errors that may arise in your code and then handle them gracefully. Begin a try\except block with the following code inserted below the code you have already written.



```
>>> import arcpy
... try:
...     arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data\\Exercise4.mdb"
...     arcpy.MakeFeatureLayer_management("Communities","Communities_copy")
...     arcpy.SelectLayerByAttribute_management("Communities_copy","NEW_SELECTION"," [COMM_NAME] = '     Tamana' ")
...     arcpy.CopyFeatures_management("Communities_copy","test")
... except:
...     print "There was a problem"
... 
```

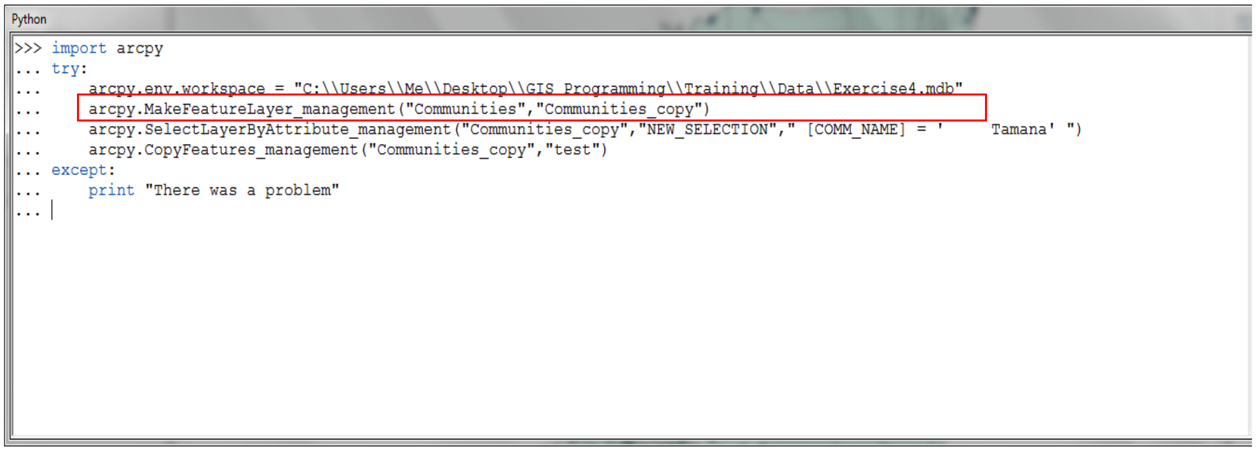
- Set the workspace for your script on a new line just below the ‘try’ statement. **Make sure you indent inside the try statement! DO NOT type the dots in the text editor.** Make sure that the path you set is directed to the location of your data.



```
>>> import arcpy
... try:
...     arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data\\Exercise4.mdb"
...     arcpy.MakeFeatureLayer_management("Communities","Communities_copy")
...     arcpy.SelectLayerByAttribute_management("Communities_copy","NEW_SELECTION"," [COMM_NAME] = '     Tamana' ")
...     arcpy.CopyFeatures_management("Communities_copy","test")
... except:
...     print "There was a problem"
... 
```

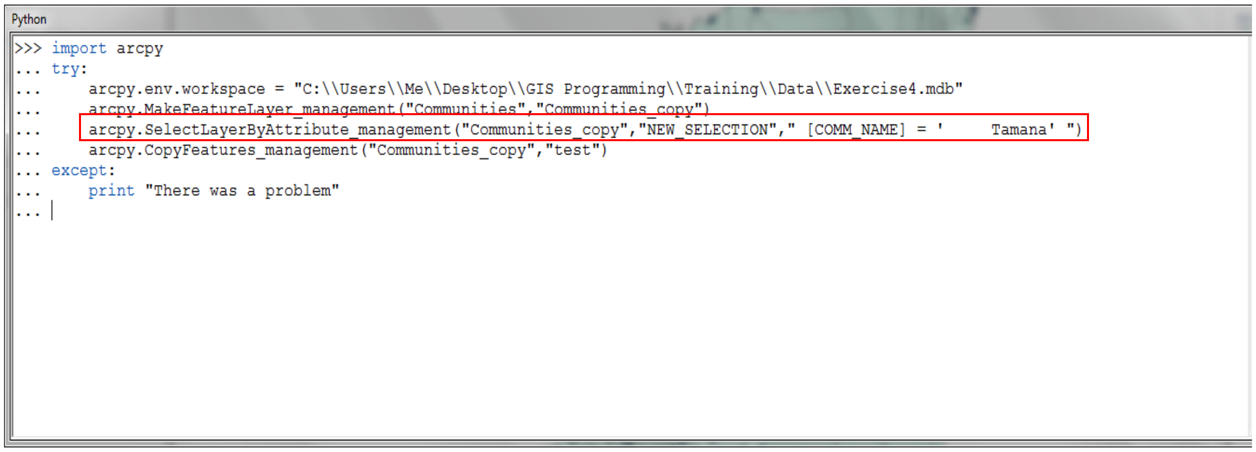
- The three tools we are going to use to perform this task are all found in the Data Management toolbox. If you haven’t already done so please open ArcGIS and we’ll examine these tools. Inside ArcToolbox click the Index tab and find and examine the following tools. Make Feature Layer Select Layer By Attribute Copy Features The first tool, ‘Make Feature Layer’, creates a temporary feature layer from an input feature class or layer file. This layer will not persist after the session ends so we’ll have to copy the records to a new, permanent file. The second tool, ‘Select Layer By Attributes’, uses a where clause to select features based on an attribute query from an input layer. In this case the input layer will be the layer we created with the first tool, ‘Make Feature Layer’. Finally, the ‘Copy Features’ tool will be used to physically copy the records that match our query to a new, permanent feature layer.

- On a new line below the line where we set our workspace, call the Make Feature Layer tool. **Make sure you indent inside the try statement!**



```
>>> import arcpy
... try:
...     arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data\\Exercise4.mdb"
...     arcpy.MakeFeatureLayer_management("Communities","Communities_copy")
...     arcpy.SelectLayerByAttribute_management("Communities_copy","NEW_SELECTION"," [COMM_NAME] = '     Tamana' ")
...     arcpy.CopyFeatures_management("Communities_copy","test")
... except:
...     print "There was a problem"
... |
```

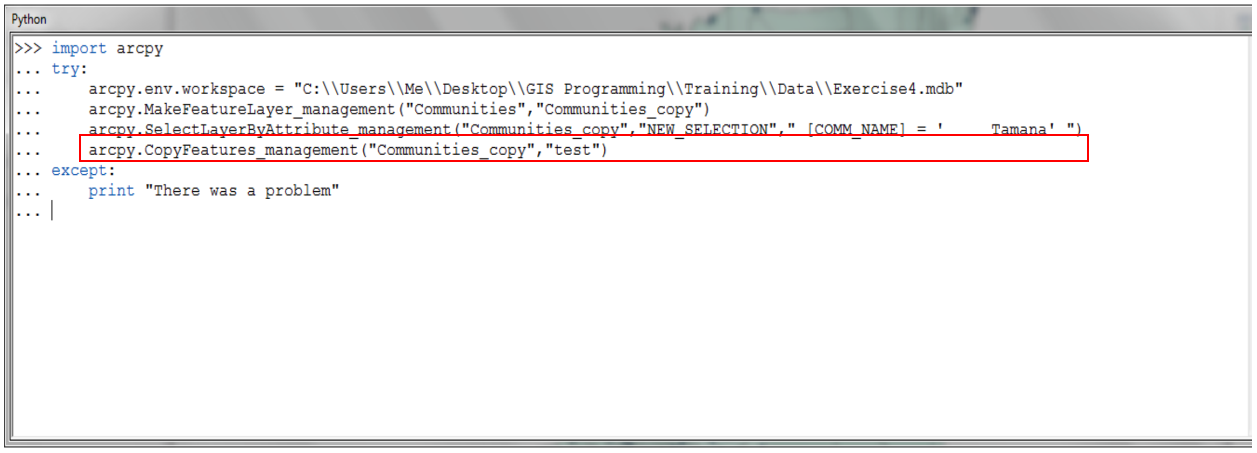
- On a new line, call the Select Layer By Attribute tool.



```
>>> import arcpy
... try:
...     arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data\\Exercise4.mdb"
...     arcpy.MakeFeatureLayer_management("Communities","Communities_copy")
...     arcpy.SelectLayerByAttribute_management("Communities_copy","NEW_SELECTION"," [COMM_NAME] = '     Tamana' ")
...     arcpy.CopyFeatures_management("Communities_copy","test")
... except:
...     print "There was a problem"
... |
```

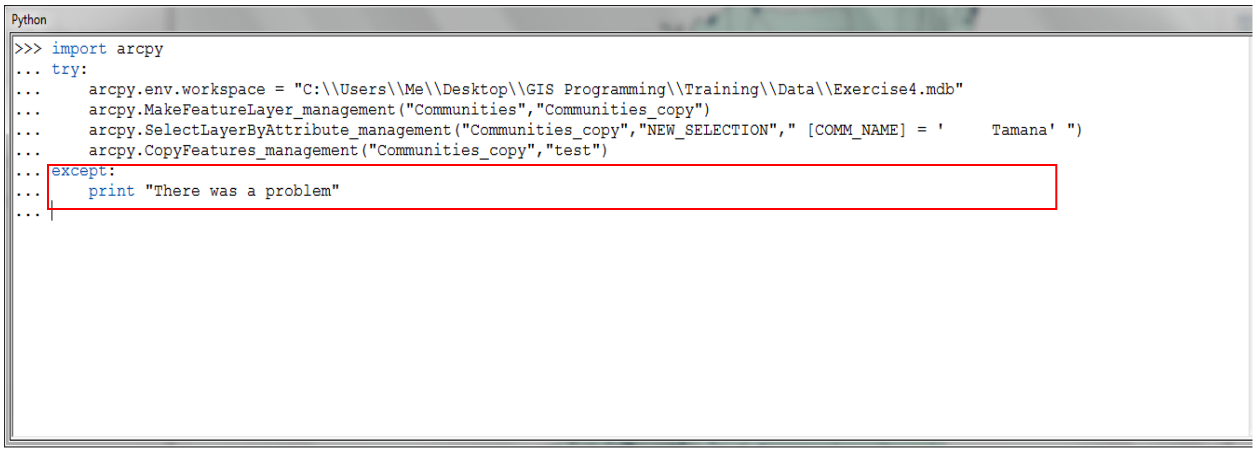
The 'Select Layer By Attribute' tool accepts a feature layer as input Communities_Copy, defines a selection type that will be performed (here we are simply creating a new selection set), and is passed a where clause that will be used to create a new selection set containing the records that matched the where clause. The where clause in this case is querying the COMM_NAME field for records that contain the string 'Tamana' which indicates the community that we are looking for. REMEMBER: Communites_Copy is a copy of the Communities layer that was created using the MakeFeatureLayer tool.

- On a new line, call the 'Copy Features' tool. This tool will create a new shapefile called 'test' which contains only those communities with the name 'Tamana'. **Make sure you indent inside the try statement!**



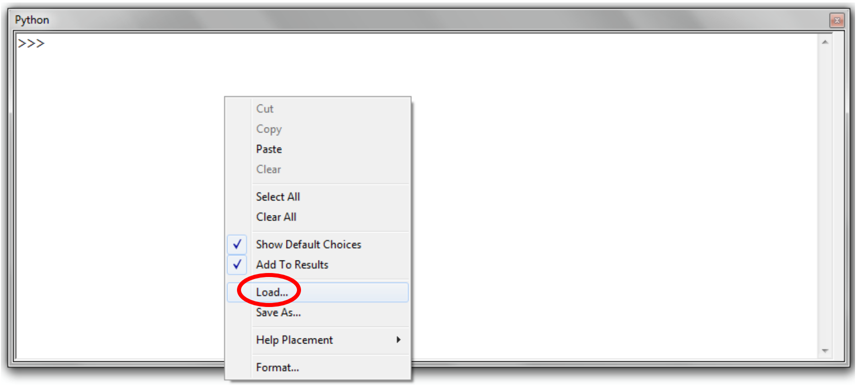
```
Python
>>> import arcpy
... try:
...     arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data\\Exercise4.mdb"
...     arcpy.MakeFeatureLayer_management("Communities","Communities_copy")
...     arcpy.SelectLayerByAttribute_management("Communities_copy","NEW_SELECTION"," [COMM_NAME] = '      Tamana' ")
...     arcpy.CopyFeatures_management("Communities_copy","test")
... except:
...     print "There was a problem"
... 
```

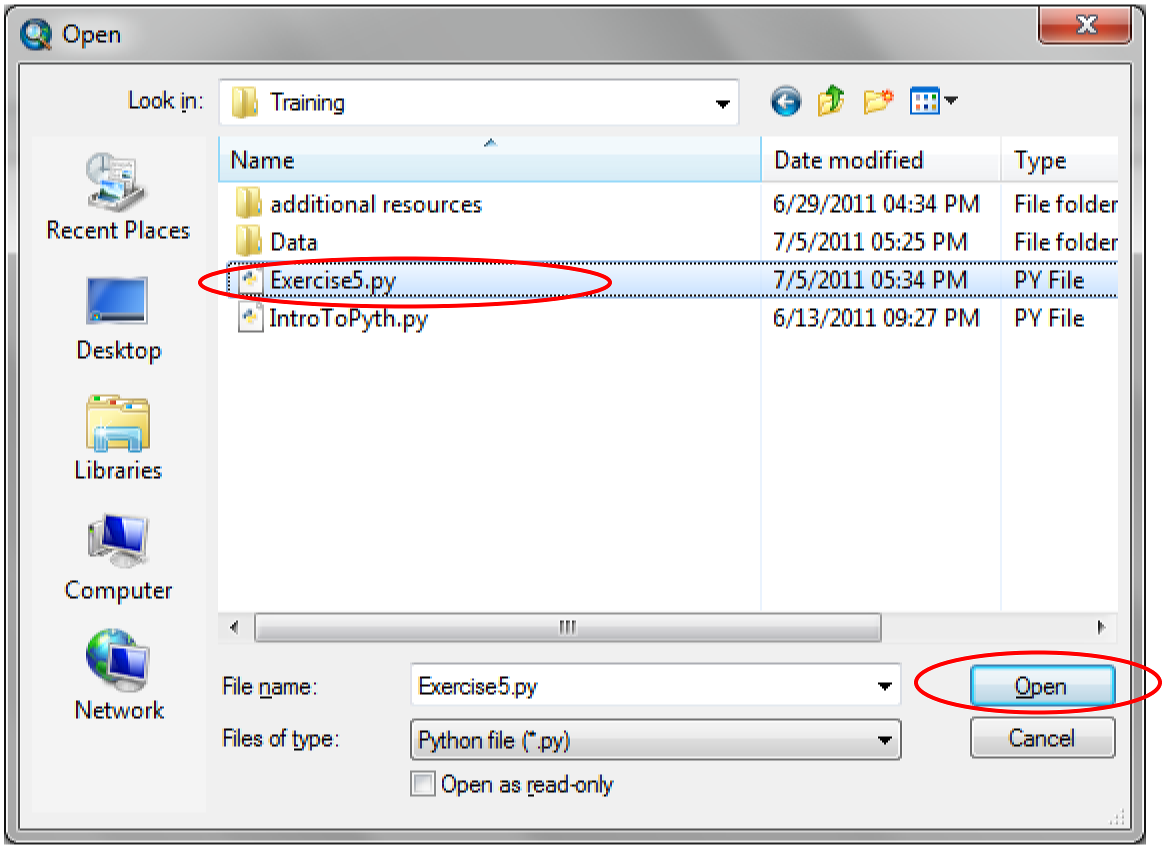
- Finally, add an 'except' block to print out a messages signifying an error if the operation failed.



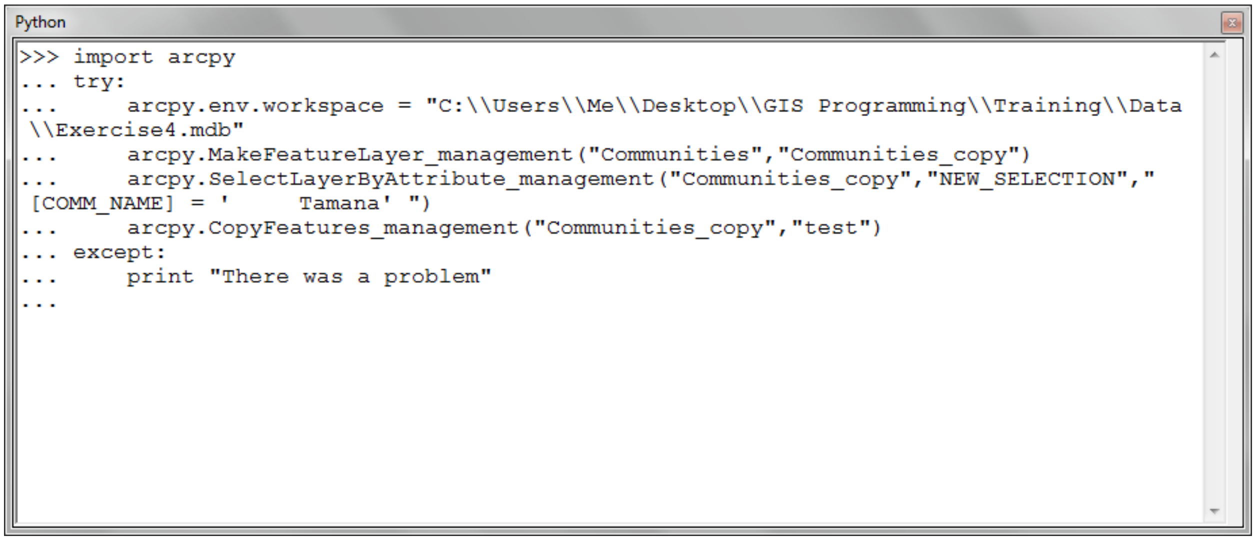
```
Python
>>> import arcpy
... try:
...     arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data\\Exercise4.mdb"
...     arcpy.MakeFeatureLayer_management("Communities","Communities_copy")
...     arcpy.SelectLayerByAttribute_management("Communities_copy","NEW_SELECTION"," [COMM_NAME] = '      Tamana' ")
...     arcpy.CopyFeatures_management("Communities_copy","test")
... except:
...     print "There was a problem"
... 
```

- Save the file as Exercise.py
- Open the Python Window in ArcGIS and right click inside the window→Load→Browse to the location of Exercise5.py on your hard disk→Select Open.



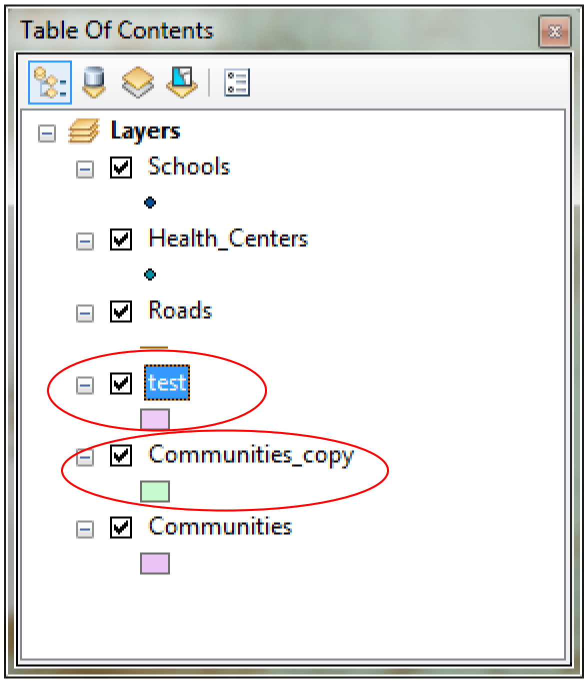


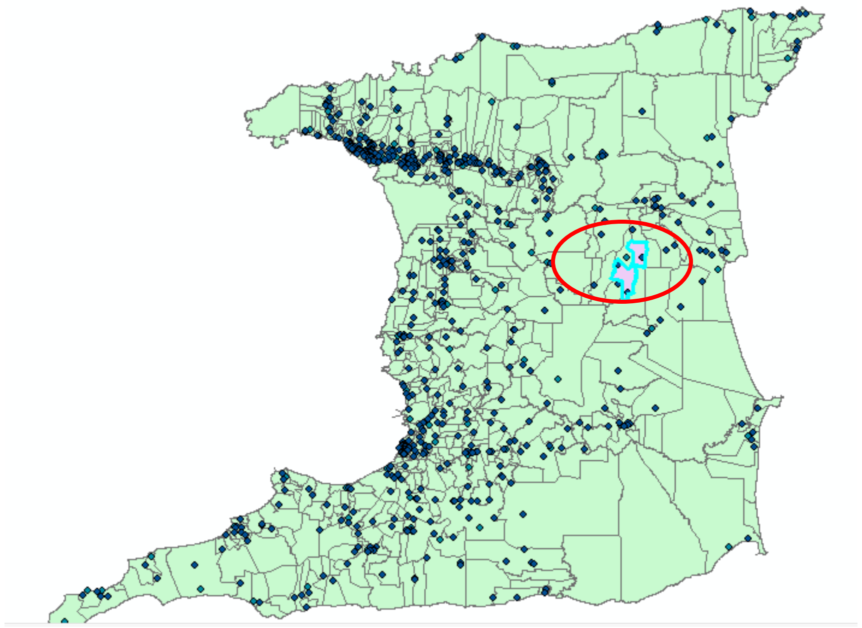
The Python code you just wrote should now be loaded into the Python Window.



```
>>> import arcpy
... try:
...     arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data
 \\Exercise4.mdb"
...     arcpy.MakeFeatureLayer_management("Communities","Communities_copy")
...     arcpy.SelectLayerByAttribute_management("Communities_copy","NEW_SELECTION","
 [COMM_NAME] = '      Tamana' ")
...     arcpy.CopyFeatures_management("Communities_copy","test")
... except:
...     print "There was a problem"
...
```

- Hit the ENTER Key to run the code. Two new files should now be created; Communities_Copy representing a copy of the original community file and test representing a subset of the communities file with only two polygons with the community name ‘Tamana’.





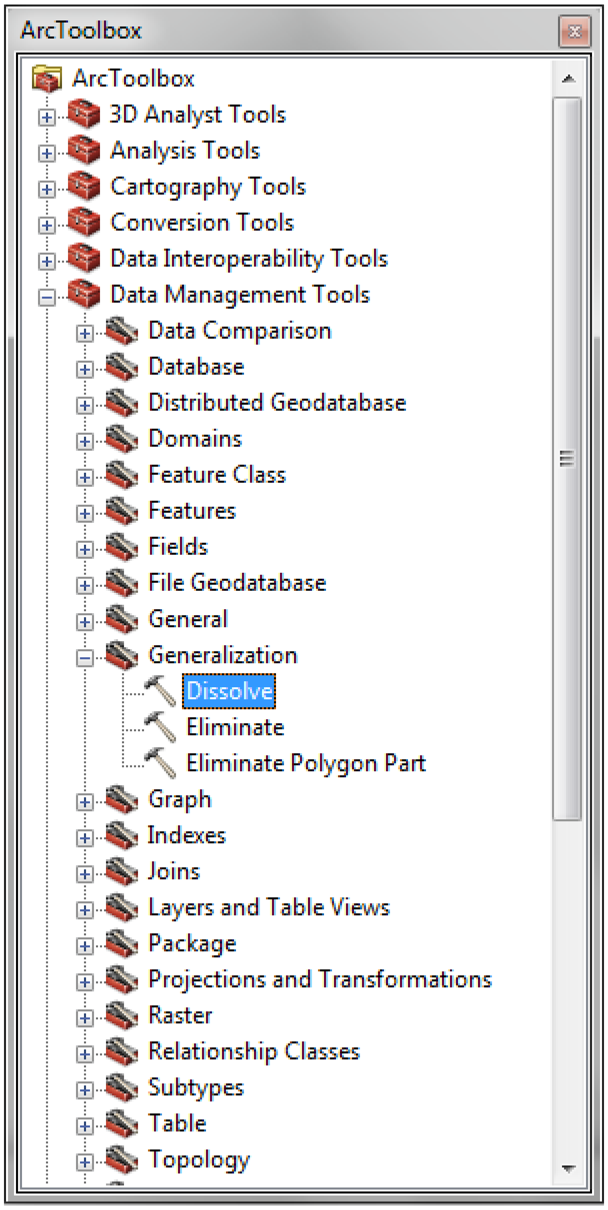
- Check the attribute table of test.shp to verify the above information.

TASK:

Write and execute Python code to **DISSOLVE** the boundaries between the polygons in test.shp into one polygon.

Hints:

1) The function you are looking for in located



2) Remember the syntax <tool name>_<toolbox alias>
3) Optional parameter can be left out for this task