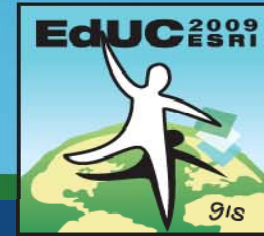


# 2009 ESRI Education User Conference

July 11–14, 2009

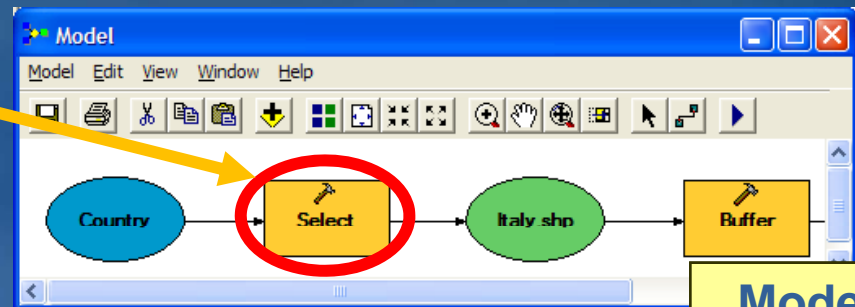
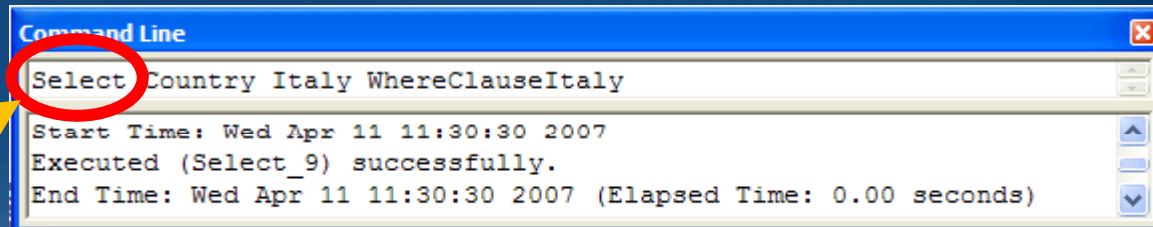
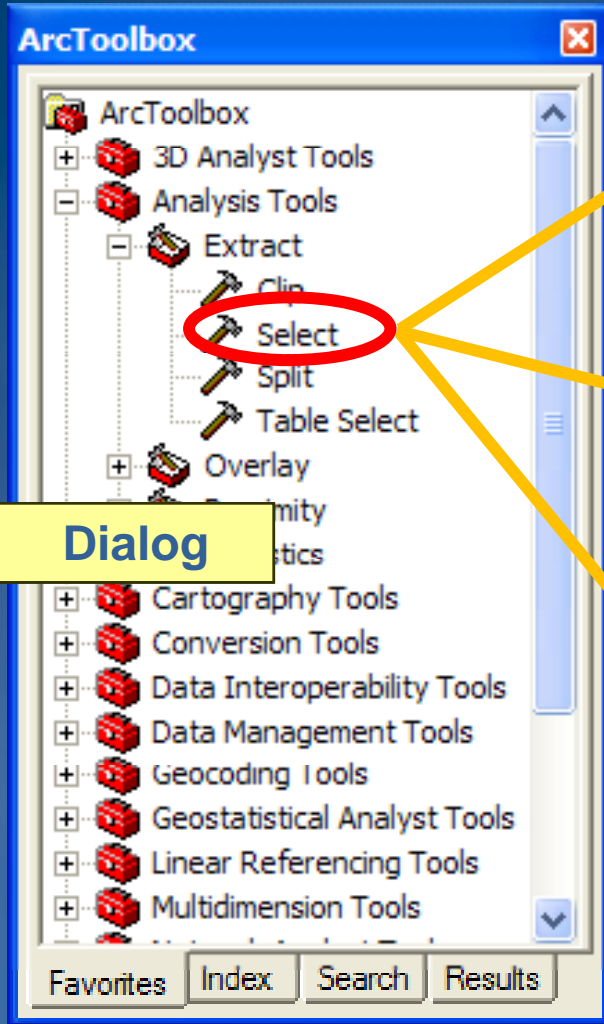


## Introduction to Geoprocessing using Python

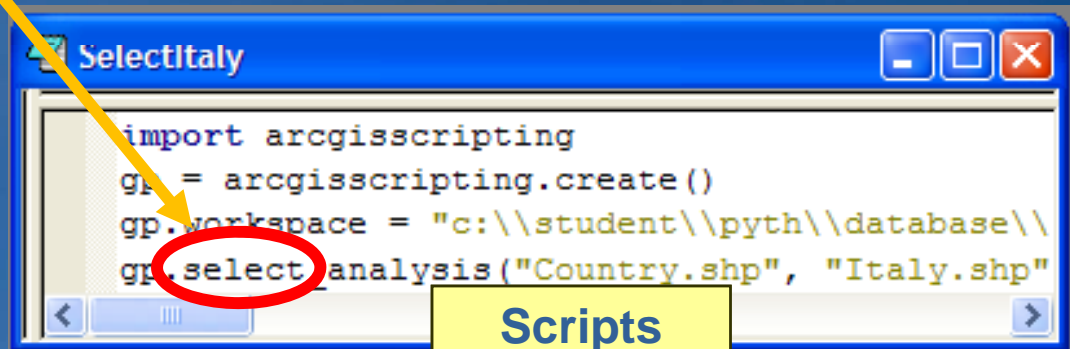
*Jorge Ruiz-Valdepeña*

# Geoprocessing options

Command line



Models



Scripts

ArcObjects

# Why write scripts?

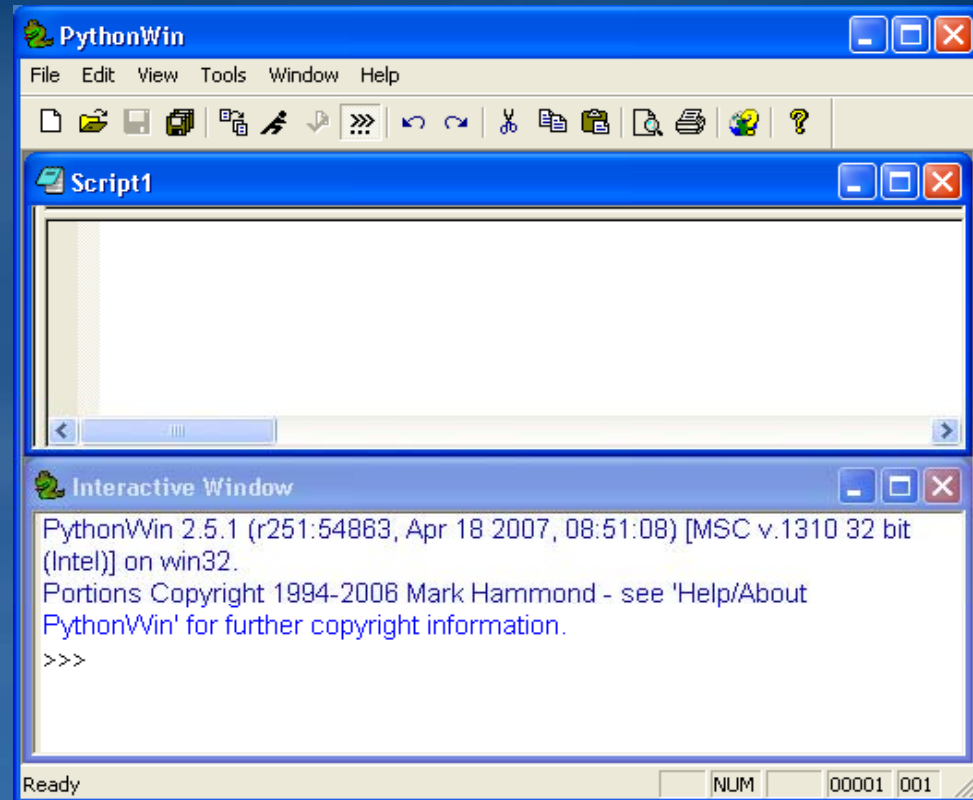
- Automate a work flow
  - Copy all incoming data into a geodatabase
  - Perform clip and buffer operations on 1000 data sets
- Run code at specific dates and times
  - Windows AT command
  - Windows scheduler
  - Every Friday print the map
- Easily distribute code
  - A script is a self-contained, single file
- Free up time for other important tasks!
- Make yourself more efficient

# Why use Python?

- **Easy to learn!**
- **Free!**
- **Open-source, object-oriented, scripting language**
  - **Can view and modify source code**
  - **Support for large projects**
  - **Easy to use**
- **Offers Development environments with debugging tools**
- **Cross platform and works in many Web-browsers**
- **Ability to compile scripts**
- **Installed with ArcGIS and ESRI samples provided**

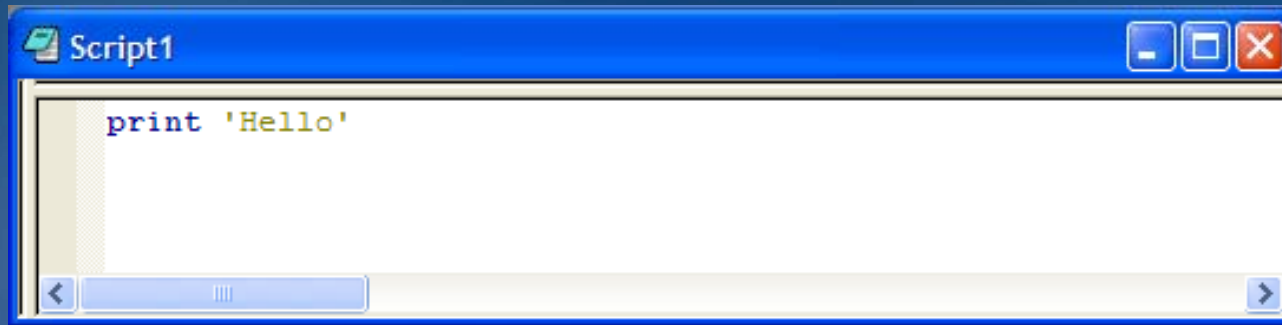
# PythonWin

- **Integrated development environment or IDE**
- **IDE contains menus, toolbars, and context menus**
  - Windows look and feel
  - All in one application
  - Script tools open in PythonWin
- **Script window** →
  - Write and save code
  - Autosave before each run
- **Interactive window** →
  - Test lines of code
  - Report messages



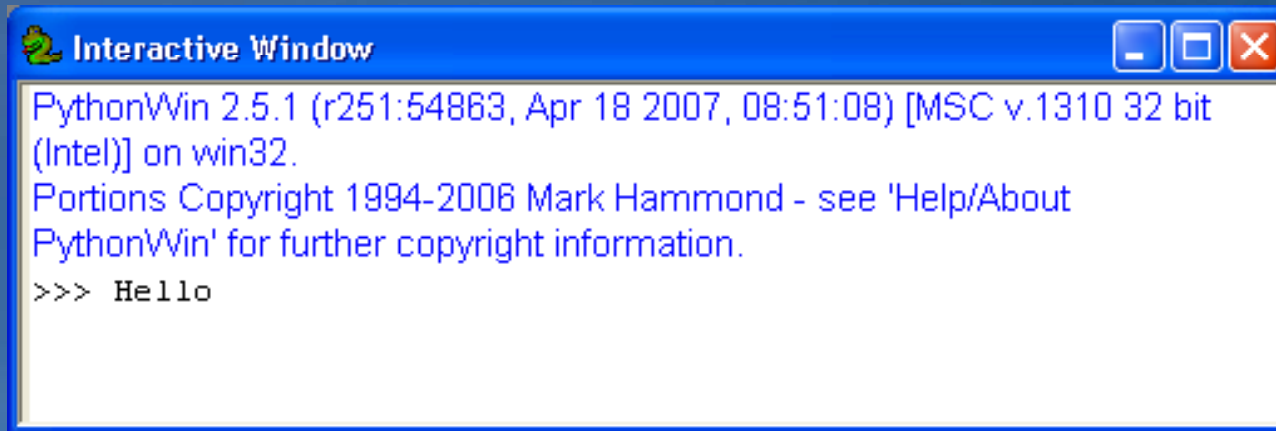
# Statements

- The print statement
- Type `print 'Hello'` in the Script window



A screenshot of a window titled "Script1" with a blue title bar and standard Windows window controls. The main area is a white text editor containing the code `print 'Hello'` on a single line. A horizontal scrollbar is visible at the bottom of the text area.

- Then click the Run button
- Prints Hello in the Interactive window



A screenshot of a window titled "Interactive Window" with a blue title bar and standard Windows window controls. The main area is a white text editor displaying the following text in blue font: `PythonWin 2.5.1 (r251:54863, Apr 18 2007, 08:51:08) [MSC v.1310 32 bit (Intel)] on win32.`, `Portions Copyright 1994-2006 Mark Hammond - see 'Help/About PythonWin' for further copyright information.`, and `>>> Hello`.

# Comments

- **Comment: A non-executable line of code**
- **Helps you remember**
- **Helps others decipher**
- **Lets you add notes**
  - One pound sign (#) for green and italicized
  - Two pound signs (##) for grey

```
# Date: August 3, 2008
```

```
# Purpose: To buffer a feature class
```

```
# This code will contain ...
```

```
# This code was written by ...
```

# Variables

- **Variables are dynamically typed**
  - No declaration keyword
  - No type assignment
  - Make up a name and set it equal to a value

```
fc = "C:\\Data\\SanDiego\\Streets.shp"
```

- **Variables are case sensitive**

```
scale = 10000
```

```
Scale = 20000
```



**Two different variables**

- **Variables can hold different data types**
  - Strings, numbers, lists, files, and more



# Decisions

- Testing conditions

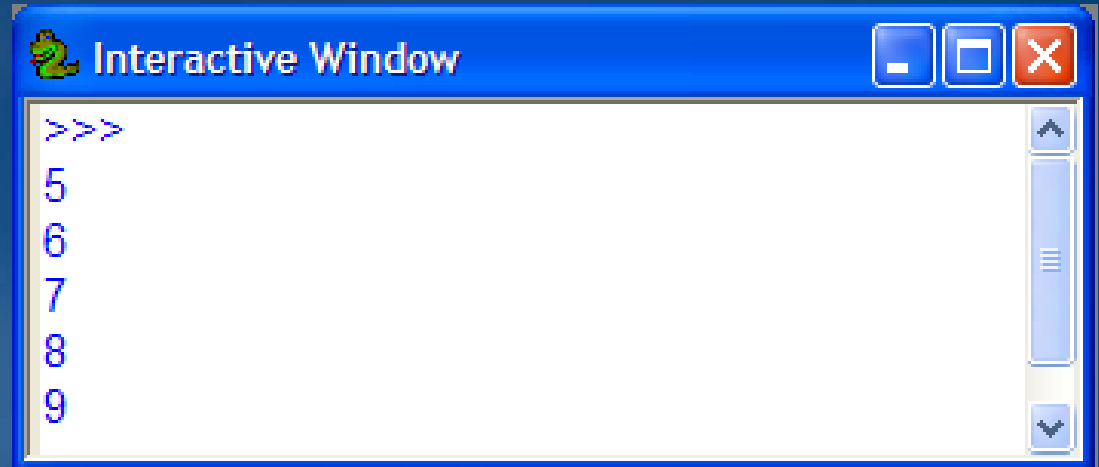
```
if x == 1:  
    print "x is 1"  
elif x == 2:  
    print "x is 2"  
else:  
    print "x is not 1 or 2"
```

- Colons used at end of each condition
- Indentation defines what executes for each condition
  - Python automatically indents when you press Enter
  - Use tabs or spaces, must be consistent
- Two equal signs for conditions, one for assignment

# Loops

- **While**

```
x = 5
while x < 10:
    print x
    x = x + 1
```



The screenshot shows a window titled "Interactive Window" with a Python logo icon. The window contains a text area with the following text: `>>>` on the first line, `5` on the second, `6` on the third, `7` on the fourth, `8` on the fifth, and `9` on the sixth. The text is in a monospaced font. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

- **Colons end each statement**
- **Indentation defines what executes**

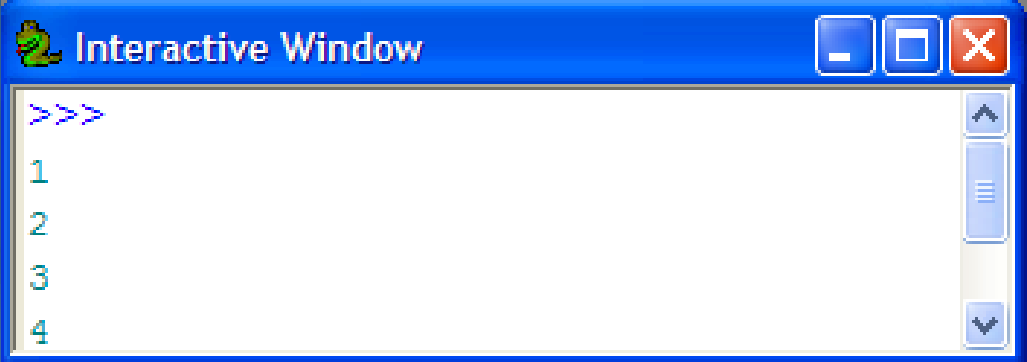
# Loops

- **While**

```
x = 5
while x < 10:
    print x
    x = x + 1
```

- **Counted**

```
for x in range(1,5):
    print x
```



The screenshot shows a window titled "Interactive Window" with a blue title bar and standard window controls (minimize, maximize, close). The window contains a Python prompt ">>>" followed by the output of a for loop: "1", "2", "3", and "4", each on a new line. The output is displayed in a monospaced font.

# Loops

- **While**

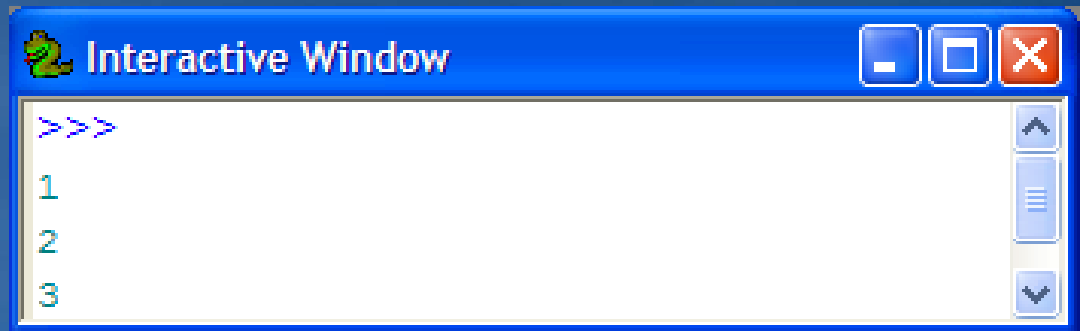
```
x = 5
while x < 10:
    print x
    x = x + 1
```

- **Counted**

```
for x in range(1,5):
    print x
```

- **List**

```
x = [1, 2, 3]
for a in x:
    print a
```



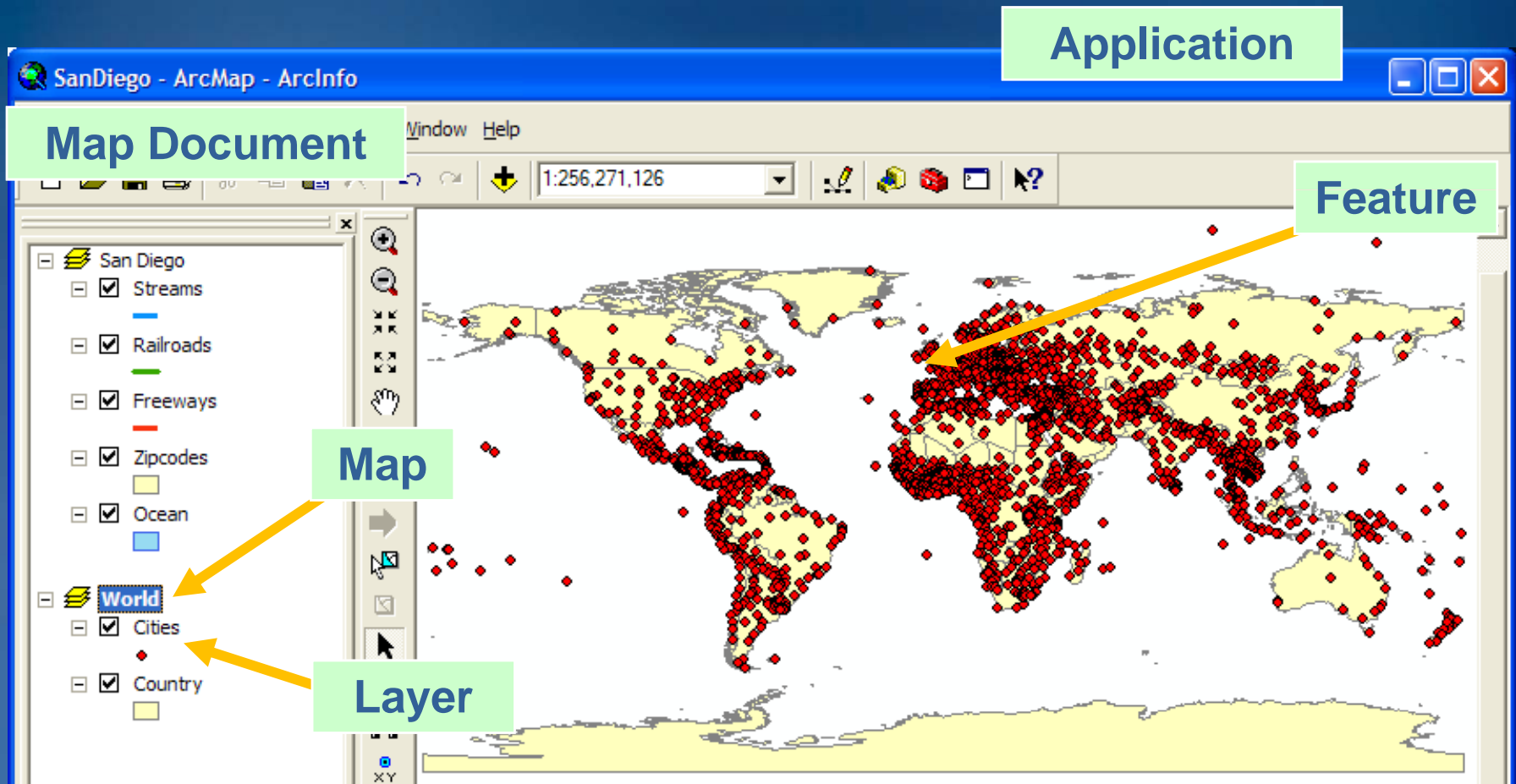
The screenshot shows a window titled "Interactive Window" with a blue title bar and standard Windows window controls (minimize, maximize, close). The window contains a text area with the following content:

```
>>>
1
2
3
```

On the right side of the text area, there are three vertical buttons: an upward arrow, a list icon (three horizontal lines), and a downward arrow.

# Common ArcObjects

- ArcGIS is built with a set of ArcObjects



# More ArcObjects

- Scripting doesn't get to User Interface objects
- Which objects can you get to?

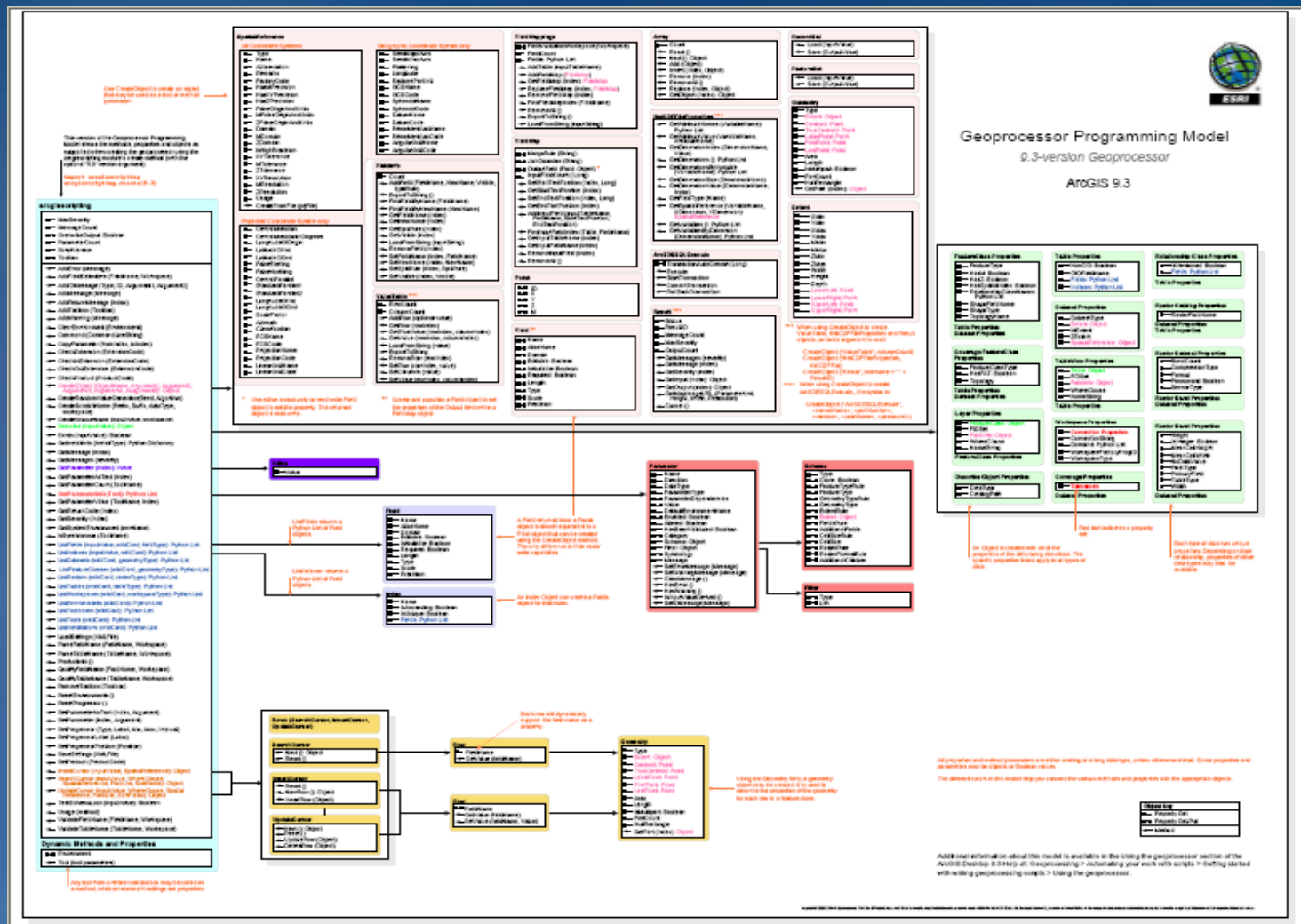
The screenshot shows the ArcMap interface with a map of North America. A table titled "Attributes of Cities" is open, displaying a list of cities. Annotations with arrows point to various elements:

- Symbol:** Points to the "Cities" layer in the Table of Contents.
- Selection:** Points to a cyan circle on the map.
- Field:** Points to the "CITY\_NAME" column in the table.
- Row:** Points to a specific row in the table.
- Feature Class:** Points to the table window itself.

FID	Shape *	CITY_NAME	CNTRY_N
569	Point	Macapa	Brazil
570	Point	Thule	Greenland
571	Point	Moosonee	Canada
572	Point	Saint John's	Canada
573	Point	Quebec	Canada
574	Point	Saint Pierre & Miquelon	France

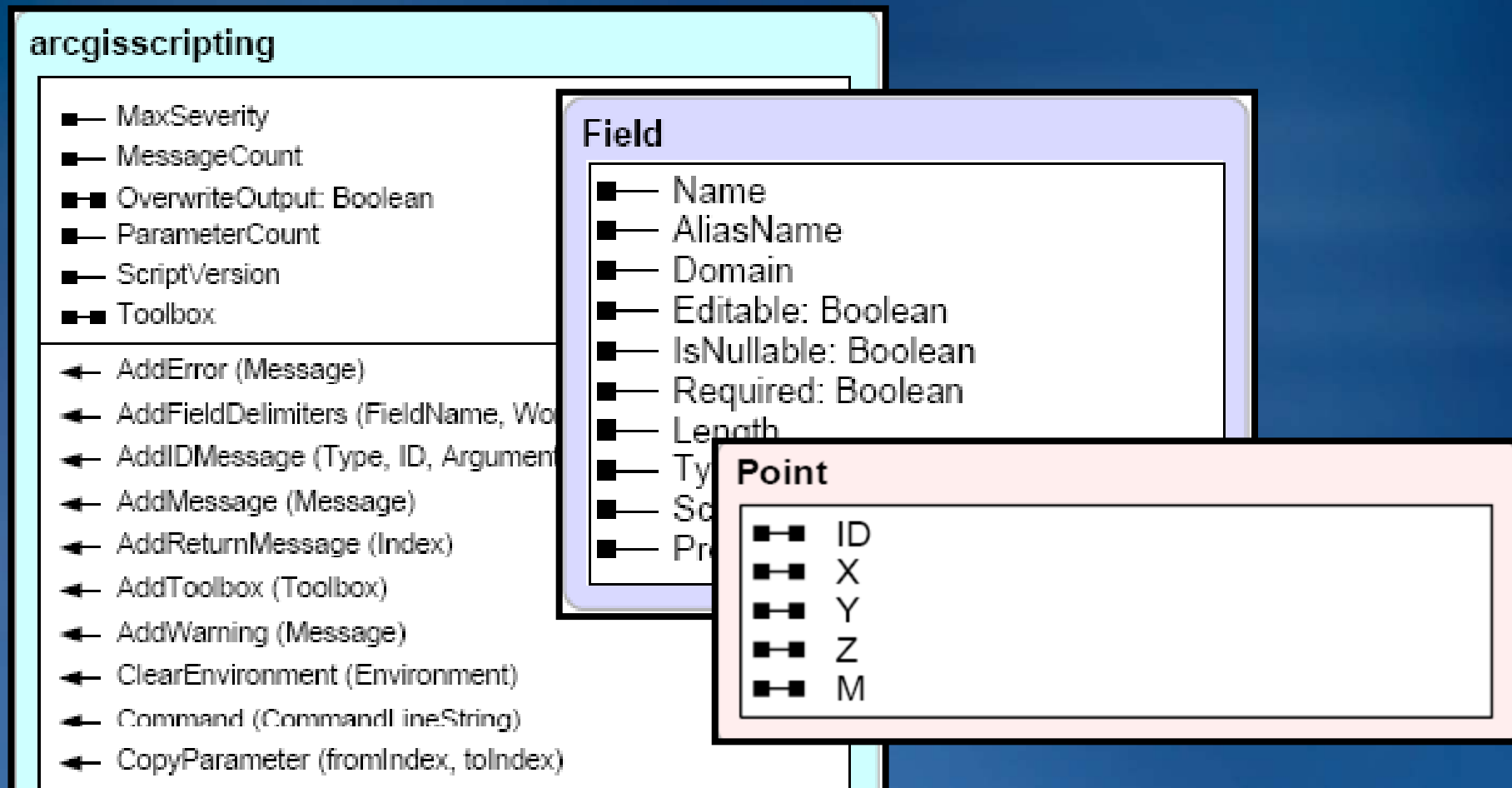
# Object Model Diagram

## Map of geoprocessing objects, properties, and methods



# Object Model Diagram

- Map of geoprocessing objects, properties, and methods





# Syntax for properties and methods

- Set a property's value

**Object.Property = Value**

```
gp.Workspace = "C:\\temp"
```

# Syntax for properties and methods

- Set a property's value

```
Object.Property = Value
```

```
gp.Workspace = "C:\\temp"
```

- Get a property's value

```
x = Object.Property
```

```
x = gp.Workspace
```

```
print "The workspace name is " + x
```

# Syntax for properties and methods

- Set a property's value

```
Object.Property = Value
```

```
gp.Workspace = "C:\\temp"
```

- Get a property's value

```
x = Object.Property
```

```
x = gp.Workspace
```

```
print "The workspace name is " + x
```

- Use a method

```
Object.Method (arg, arg, ...)
```

```
gp.Buffer_analysis (fc, "C:\\temp\\buff.shp", 100)
```

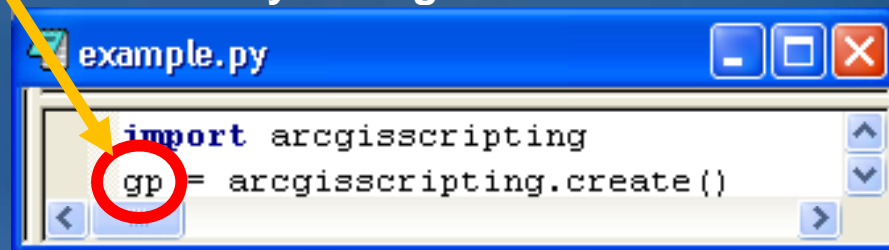
- Parentheses around arguments
- Arguments separated by commas

# Create a geoprocessor object in code

- Geoprocessor can be used in any COM language
  - Component object model language
  - Perl, VBScript, JScript, Python, VBA, VB, and C#

- **arcgisscripting module**
  - ESRI-written
  - Cross-platform

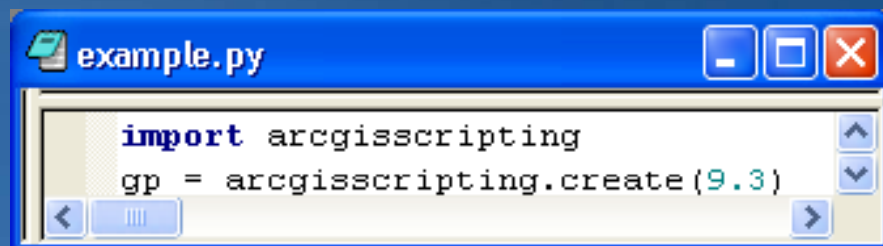
You access all geoprocessing functionality through this variable



```
example.py
import arcgisscripting
gp = arcgisscripting.create()
```

- **9.3 Argument**

- accept and return common Python structures such as lists and Booleans

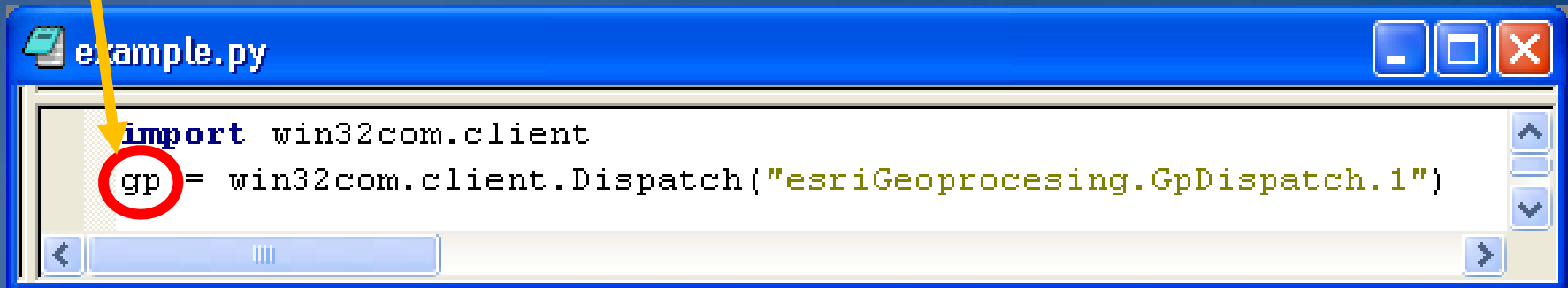


```
example.py
import arcgisscripting
gp = arcgisscripting.create(9.3)
```

# Create a geoprocessor object in code in previous versions

- The Python win32com module must be loaded using the Import command.
  - This module enables the COM IDispatch communication within Python.

You access all geoprocessing functionality through this variable



```
example.py
import win32com.client
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")
```

The screenshot shows a window titled 'example.py' with a blue title bar. The code inside is: `import win32com.client` and `gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")`. A red circle highlights the variable `gp` in the second line, and a yellow arrow points from the text above to this circle. The window has standard Windows window controls (minimize, maximize, close) in the top right corner and a scrollbar at the bottom.

# The Buffer tool

- **Syntax**

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

- **Example**

```
gp.Workspace = "C:\\Database\\World"  
gp.Toolbox = "Analysis"  
gp.Buffer("Lakes.shp", "BuffLakes.shp", "100 feet")
```

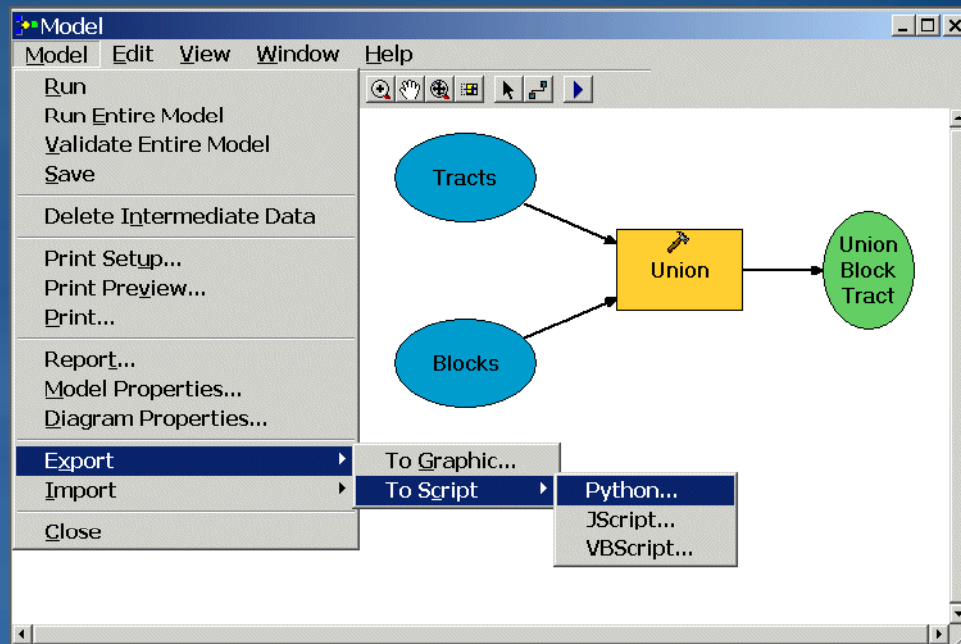
- **Notes**

- If units not specified, input feature class units used
- If specifying units, make the argument a string

# Syntax

If I want to use the UNION tool, how would I know that the inputs are separated by semicolons?

1. ArcGIS Desktop Help
2. If help is not detailed enough, export the tool from a model to a script



# Multiple tools

- Run many tools in succession
- Use one tool's output as another's input

```
# Find a location for a new hotel that must be > 10000 ft  
# from an existing one and within 2000 ft of a freeway.
```

```
gp.Workspace = "C:\\Database\\SanDiego.mdb"
```

```
# Buffer Freeways
```

```
gp.Buffer_analysis("Freeways", "BuffFreeways", 2000)
```

```
# Select the Inns
```

```
gp.Select_analysis("MajorAttractions",  
                  "HolidayInns", "[NAME] LIKE 'HOLIDAY INN*'" )
```

```
# Buffer the Inns
```

```
gp.Buffer_analysis("HolidayInns", "BuffHolidayInns",  
                  10000)
```

```
# Erase the buffered Inn areas from the Freeways buffer
```

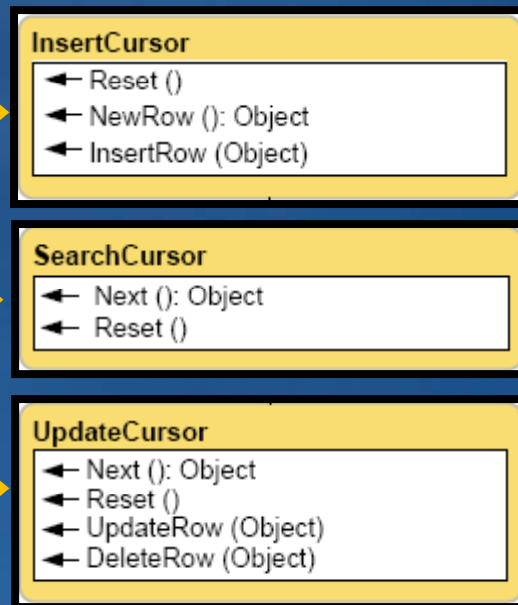
```
gp.Erase_analysis("BuffFreeways", "BuffHolidayInns",  
                  "SuitableAreas")
```



# Cursors

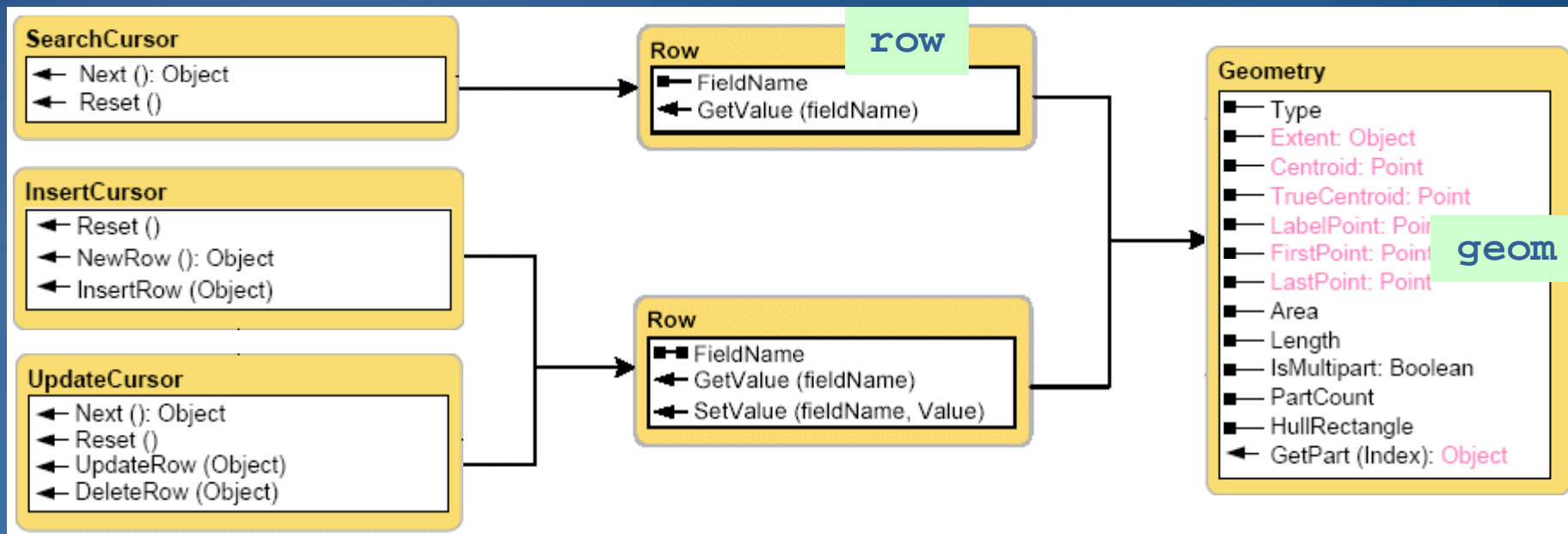
- Get or edit rows and values in a table
- `SearchCursor` reads values in a row
- `UpdateCursor` makes changes to row values and deletes rows
- `InsertCursor` is used to insert new rows

← `InsertCursor (InputValue, SpatialReference): Object`  
← `SearchCursor (InputValue, WhereClause, SpatialReference, FieldList, SortFields): Object`  
← `UpdateCursor (InputValue, WhereClause, Spatial Reference, FieldList, SortFields): Object`



# Geometry

```
cur = gp.SearchCursor ("Climate")
row = cur.Next()
while row:
    geom = row.Shape
    print geom.Area
    print geom.Extent
    row = cur.Next()
```



# Resources

- Web site: [www.python.org](http://www.python.org)
  - Tutorials, Documentation, Forums
- Python books
  - Learn to Program Using Python, by Alan Gauld
  - Learning Python (2<sup>nd</sup> Edition), by Mark Lutz and David Ascher
  - Python Essential Reference (2<sup>nd</sup> Edition), by David M. Beazley
  - The Quick Python Book, by Daryl Harms and Kenneth McDonald
  - Python Programming on Win32, by Mark Hammond and Andy Robinson

## Get a free 45-minute hands-on lesson at the **Hands-On Learning Center**

Topics include:

- **Introduction to ArcGIS Desktop**
- **Creating a Map In ArcGIS**
- **Basics of the Geodatabase Model**
- **and more**

Location: **ESRI Showcase**