A Guide to the Python Universe for ESRI Users

Howard Butler

Center for Survey Statistics and Methodology

Iowa State University

Ames, IA 50010

Abstract

ArcGIS 9 will provide scripting capabilities with the Geoprocessing Framework. This will allow scripting languages like Python to automate tasks with ArcGIS. Python has historically provided many other opportunities for scripting GIS. This paper describes an overview of the GIS tools available for use scripting with Python. Functionality such as working with grids, image manipulation, reading and writing shapefiles, delivering maps to the Web, and communicating with the Microsoft TerraServer are described. An overview will give the reader more tools in his/her toolbox to integrate with existing software.

Introduction

Scripting in ESRI environments historically has belonged to two schools of thought. First is the AML (Arc Macro Language) model, which wears its PrimOS heritage on its sleeve. You pipe output to files, data handling is file-system and directory-based, and code is very linear in nature. The other school is demonstrated in Avenue, which wears a Smalltalk badge. Object.request is the name of the game, things don't have to be linear, I/O is sometimes a struggle, and integrating with other programs is a mixed-bag. Both are custom languages that have their own dark, nasty corners.

With the introduction of ArcGIS 8, your scripting-based view of the world was turned upside down. Interface-based programming required you to use a "real" programming language like C++ or Visual Basic to access the functionality that Arc 8 provided. There was no longer such a thing as a "script" that allowed you to automate a series of tasks. Instead, you had to write your own executables, navigate a complex tree of interfaces and objects to find the tools you needed, and compile DLLs and type libraries to expose your own custom functionality.

Now, with the introduction of ArcGIS 9, ESRI has once again provided access to their software with scripting capabilities. They realized that many of their users don't want or need to be programmers, but would rather have some tools at their disposal to solve the problem at hand. These tools include nice, consistent GUIs; scriptable objects; and nuts and bolts programming tools necessary for customization.

To fulfill this need, ESRI has chosen to support a variety of scripting languages using ArcObjects – starting with the GeoProcessing Framework. One of the languages supported is Python, an Open Source, interpreted, dynamically-typed, object-oriented scripting language.

The rest of this article will focus on giving you an overview of what is available in the Python universe to help you with GIS programming and integrating with ESRI tools.

Introducing Python

Python was first released in 1991 by Guido van Rossum at CWI in the Netherlands. Yes, it is named after Monty Python's Flying Circus, which Guido loves. Its name also means that many references from the movies and the show are sprinkled throughout examples, code, and comments. Many of Python's features have been cherry-picked from other languages such as ABC, Modula, LISP, and Haskel. Some of these features include advanced things like metaclasses, generators, and list comprehensions, but most programmers will only need the basic types like lists, dictionaries, and strings that Python provides to get going.

Although it is almost thirteen years old, Python is currently at release 2.3. This reflects the design philosophy of the Benevolent Dictator For Life (Guido) and the group of programmers that continue to improve Python. They strive for incremental change, attempting to preserve backwards-compatibility, but they redesign areas seen in hindsight as mistakes when necessary.

The Design of Python

Python is designed to be an easy-to-use, easy-to-learn dynamic scripting language. What this means for the user is that there is no compiling (the language is interpreted and compiled on-the-fly), it is interactive (you can bring up the interpreter prompt much like a shell and begin coding right away), and it allows users to learn its many layers of implementation at their own pace.

The design philosophy of Python was most clearly described by Tim Peters, one of the lead developers of Python. A Python programmer can use these maxims to help guide them through the language and help them write code that could be considered *pythonic*.

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
*Although never is often better than *right* now.*
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

Installation

You can download a copy at http://www.python.org, but if you have ArcGIS 9, it should already be installed. On the Windows front, you'll also want to install win32all, an extension for Python that provides access to the Win32 API. Download the appropriate copy for your install of Python at http://starship.python.net/crew/mhammond/win32/Downloads.html. A complete distribution of Python with the Windows extensions and extended documentation is also available from ActiveState http://www.activestate.com/Products/ActivePython/.

Once you have the basic pieces in place, you can install other packages that extend the capabilities of Python. The most common ones include NumPy http://www.pfdubois.com/numpy/ (efficient array handling), and PIL http://www.pythonware.com/products/pil/ (image manipulation). Python's philosophy is to stand on the shoulders of giants, and incorporating extra functionality is usually as simple as finding the appropriate package and installing it.

Python and GIS

Python provides many opportunities for integration within GIS computing systems. Cross-platform capabilities and ease of integration with other languages (C, C++, Fortran, and Java) means that Python is most successful in the gluing of systems together. Because of the fluid language design, the development of large-scale applications is also easily supported.

There are many libraries and tools already developed for working with GIS data in Python. The basics are covered, including the manipulation of shapefiles, grids, and images, as well as more sophisticated stuff like scripting of SDE and interaction with web services and databases.

Vector formats

A Python wrapper of the Open Source library Shapelib (http://shapelib.maptools.org) called pyshapelib is available for working with shapefiles. You can download it at (http://www.hobu.biz/software/pyshapelib). It provides access to the individual vertices of the shape, access to the dbf file, and simple shape indexing. This library is useful if you want to manipulate the raw geometry of a shapefile or pan through the dbf file to get to specific records.

A Python wrapper of the Open Source library OGR (http://gdal.maptools.org/ogr/) called OGR is available for working with vector formats other than shapefiles. These formats include MapInfo, ArcInfo coverage, PostGIS, Oracle Spatial, TIGER, SDTS, OPeNDAP, DGN, and Microstation DGN. OGR is part of the GDAL library, and it can be downloaded with the GDAL distribution (http://hobu.stat.iastate.edu/gdal-1.2.0win32.exe)

Grids

A Python wrapper of the Open Source library Geodata Abstraction Library (GDAL) (http://gdal.maptools.org) is available for working with ArcInfo grids. A multitude of raster formats are supported by GDAL including, JPEG2000, BSP, USGS DEM, Military Elevation Data, ECW, GRASS, TIFF/GeoTIFF, NetCDF, Imagine, and SDTS. The Python library is available for download for Windows at (http://hobu.stat.iastate.edu/gdal-1.2.0win32.exe). There are many other formats available that are not listed here.

GDAL in combination with Numeric Python gives you the flexibility to write your own map algebra operations using whichever format suites your need. You could write a process that lived on a webserver, downloaded data with WCS (OGC's Web Coverage Service), processed some algebra, and delivered an image to the web browser, for example. The possibilities are endless once you have the ability to divorce the processing of data from their display.

An example

Say you wanted to find the average value of a grid across both the rows and the columns. The data are in ArcInfo binary format, and they are integer data. Using the interactive Python window, first import the GDAL library. Then tell GDAL where to find the .adf of the grid that you want to open (using Python's raw mode to input the string). Pass the contents of the grid into a Numeric Python array, and then use Numeric's processing methods to produce the average.

```
>>> import gdal
>>> gd =
gdal.gdal.Open(r'E:\gis\US_Elevation\usdem_2k\w001001.adf')
>>> array = gd.ReadAsArray()
>>> avg = Numeric.average(Numeric.ravel(array))
>>> avg
-0.0071967281963325313
```

Projections

A Python wrapper of the Open Source library Proj.4 (http://proj.maptools.org) called py-Projection (http://hobu.biz/index_html/software/pyprojection/) is available for projecting coordinates from one projection to another. Proj uses the EPSG code system, although you can define your own projections by using raw parameters. The use of Proj is as simple as in defining the current projection, the X and Y coordinates, and calling a method that transforms them to the desired projection.

An Example

```
import Projection

albers = ["proj=aea",
          "ellps=GRS80",
          "datum=NAD83",
          "lat_1=29.5",
          "lat_2=45.5",
          "lat_0=23.0",
          "lon_0=-96.0",
          "x_0=0.0",
          "y_0=0.0"]

p2 = Projection.Projection(albers)

print '----------Albers--------------'
print 'Location: -93.00W, 42.00N'
print "Forward: ", p2.Forward(-93.00, 42.00)
print "Inverse: ", p2.Inverse(0.0, 0.0)
```

SDE

I have developed a Python wrapper of the ESRI SDE C API called PySDE (http://hobu.stat.iastate.edu/pysde) that is available for writing scripts that manipulate SDE. Almost all of the SDE C API is wrapped and has corresponding methods in Python that you can call. PySDE is Open Source, but you will need a licensed copy of the SDE C API to be able to use it.

I developed PySDE because I felt there was a need to have the ability to prototype and script the SDE engine. I wanted lean scripts that ran on unix-like platforms without the requirement of bringing up ArcGIS to process data. I have used PySDE to develop a specialized geometry algebra engine, administration scripts (drop this table, clean up log files, copy this data, etc.), and many data manipulation scripts. Another advantage of programming with PySDE is the immediacy of the Python interactive window – you can type in commands and see their effect in real time. This is a real time saver when navigating complex hierarchies like the SDE C API.

Web GIS and Python

Python is perfectly suited for web development. In my experience, web development with Python is often much faster than technologies such as Java or .NET. There are many tools available to aid you when doing web development. These include application servers such as Zope (http://www.zope.org), maprendering servers such as MapServer (http://mapserver.gis.umn.edu), and network protocol layers such as Twisted (http://twistedmatrix.com/products/download). This section will describe some of the common Open Source tools for web programming in Python with respect to GIS.

Web Services

Web Services (SOAP, XMLRPC, and REST-ful clients) are all the rage these days. Web Services allow you to encode an XML-structured request to a server and have it respond back with XML-structured data. This architecture allows you to more easily separate the data storage and management portion of your system from the application side of the fence. Python provides many tools for working with Web Services, with XMLRPC built right into the language, and many additional libraries available for working with SOAP (http://pywebsvcs.sf.net) and REST.

pyTerra

One GIS Web Service that is quite useful is the TerraService SOAP API. I have developed a Python wrapper for the TerraService SOAP API called pyTerra (http://hobu.stat.iastate.edu/pyTerra) that makes it very easy to interact with. Say, for example, you would like to find the DOQ date of a specific longitude and latitude. One way would be to dig through the tapes, find the FGDC metadata, open it in a reader (maybe ArcCatalog), and record the value. While this method works, it sure is not scalable, and if you had to look up the image dates for ten or fifteen thousand points, a program is the only way to do it.

Fortunately, the TerraServer stored the imagery acquisition dates along with the image data. You can easily use the Web Services API that pyTerra provides to quickly get at this information and do whatever you need with it. Here is an example that gets the DOQ and the DRG mapsheet date from TerraServer

```
>>> from pyTS import TerraImage
>>> from pyTS import pyTerra
>>> apt = TerraImage.point(42.00, -93.00)
>>> drg = pyTerra.GetAreaFromPt(apt, 'Topo','Scale64m', 1, 1)
>>> doq = pyTerra.GetAreaFromPt(apt, 'Photo','Scale64m', 1, 1)
>>> drg.Center.TileMeta.Capture
'1976-07-01T00:00:00.0000000-07:00'
>>> doq.Center.TileMeta.Capture
'1994-04-18T00:00:00.0000000-07:00'
```

From this traceback of the Python interactive window, you can see that the DRG and DOQ dates are 1974 and 1994, respectively. Once you have these strings, you could parse them into dates and insert them in your database, or you could capture the date information and use it to burn it into a map image using the Python Imaging Library (PIL). Python makes it easy to work with Web Services, and tools like pyTerra are available to do much of the heavy lifting for you.

Python Books

There are many books out on the market to give you a general Python programming background. Some of the best, in my opinion, are from O'Reilly, but there are others from publishers such as New Riders and Apress that will give you a good introduction. This section gives a short review of each of the books along with a description of its topic.

Learning Python
Mark Lutz and David Ascher
O'Reilly

At this point, Learning Python is probably the most complete book for an introduction to Python, especially if you are coming from languages such as Visual Basic and Avenue. The examples and descriptions in this book are precise, relevant, and clear. I find myself going back to this book frequently, even though I have been writing Python since 1999. It teaches the basics well, and it shows you how to write *pythonic* code.

Python Essential Reference
David M. Beazley
New Riders

Python Essential Reference is truly a reference book. If you are already a proficient coder in another language, I would choose this book over *Learning Python*. It has everything you need and is very terse. A new edition should be in the works soon, as this book mainly references Python 2.1, which ArcGIS 9 ships with.

Programming Python
Mark Lutz
O'Reilly

Programming Python aims to be the equivelant of the "camel" book in the Perl world. The updated second edition balloons to over 1000 pages, and there is plenty of choice bits in there for you to devour. Of all the books, it has the best coverage of using the Python C API (which shouldn't go out of date too quickly now that it has been updated for Python 2.2).

My criticisms of this expensive book would be that it is too verbose, is targeted too clearly at unix programmers, and focuses on larger example applications to teach. Also, because it is so large, holding the book in your lap is problematic. I mainly use it as a C reference and look to other books when I need help with more specific things.

Python in a Nutshell
Alex Martelli
O'Reilly

Python in a Nutshell is likely the most up-to-date, complete, and most poetic Python book available. I had the pleasure of eating lunch with the author, Alex Martelli, at the 10th Annual Python Conference. He is a very articulate speaker, and this carries over to his writing as well. This book covers the breadth of Python. Each chapter covers a separate problem domain, and gives a great overview (with great detail) of what is possible with Python.

Python Cookbook
Alex Martelli and David Ascher
O'Reilly

Python Cookbook is a book that was written by the users of Python. A website (http://aspn.activestate.com/ASPN/Python/Cookbook/) was developed (outlined in this O'Reillynet article http://www.onlamp.com/pub/a/python/2002/08/01/cookbook.html) where users submitted recipies of how they solve problems with Python. The authors then took the recipes, organized them into chapters, and added coherence to the thing.

There are some real gems in here, especially in the algorithms chapter that was edited by Tim Peters. It also gives a good overview of how people express problems in Python, how they solve them, and tips that you can use to make your life easier. The book conveys the community of Python as well as its problems. I recommend this book to be your third or fourth Python book to get after you've covered the basics.

Python Programming on Win32
Mark Hammond and Andy Robinson
O'Reilly

Python Programming on Win32 will be a useful book to someone using the Geoprocessing scripting engine in ArcGIS 9. It almost exclusively covers using the author's Python COM extensions for Windows. It covers general usage of COM, how to script using the IDispatch interface exposed by Excel, and how to work at the systems level with users, groups, and files.

I also expect that there will probably be a second edition of this book in the near future (likely to happen before the release of ArcGIS 9), as the COM extensions for Python have changed plenty in the three years since this book was written.

Text Processing in Python
David Mertz
Addison-Wesley

Text Processing with Python covers state machines, regular expressions, and esoteric topics. Unless you are doing something that needs these, avoid purchasing this spendy book. The author also writes an ongoing series at IBM DeveloperWorks called Charming Python that is well-written and covers advanced Python topics.

Jython Essentials
Samuele Pedroni and Noel Rappin
O'Reilly

Jython is a version of Python that runs on the Java Virtual Machine. It gives you native access to Java classes, and it allows you to keep the productivty of Python in a Java environment.

The book covers the basics, but also includes things like using Jython in a JSP environment, which I found very handy for doing ArcIMS development. I was able to develop my IMS maps much quicker using Jython than equivalent straight JSP, and I recommend it if you find yourself in a similar situation.

Online Resources

Documentation and online articles are probably the best way to stay abreast of updates to Python software, new techniques and methods, and new libraries that add capabilities to the language.

Python Newbies Page	http://www.python.org/doc/Newbies.html
Python How-tos	http://py-howto.sourceforge.net/
O'Reilly Python DevCenter	http://www.onlamp.com/python/
Daily Python	http://www.pythonware.com/daily/
Python Beginners' Mistakes	http://zephyrfalcon.org/labs/beginners_mistakes.html
Dive Into Python	http://diveintopython.org/
Thinking in Python	http://www.mindview.net/Books/TIPython
Data Structures and Algorithms with Object-Oriented Design Patterns in Python	http://www.brpreiss.com/books/opus7/
Beginning Python for the Statistician	http://www.public.iastate.edu/~rdecook/mywebsite/Python_tutorial.pdf

Conclusion

Python can provide you with a complete set of tools for your GIS toolbox. In combination with ArcGIS, the possibilities are endless. New technologies such as Web Services as widely supported in Python, and there are many online and paper resources to help you when developing Python.

A companion to this paper is available at my website at http://hobu.biz/software/python_guide_esri/

Universe: a software platform for measuring and training an AI's general intelligence across the world's supply of games, websites and other applications. Navigation. Project description.Â Additionally, some environments include a reward signal sent to the agent, to guide reinforcement learning. Weâ€™ve included a few hundred environments with reward signals. These environments also include automated start menu clickthroughs, allowing your agent to skip to the interesting part of the environment. Jupyter Notebooks have become a crucial tool in the Python and Data Science communities over the past years. Their seamless integration with some of the most important Python libraries and theirâ€¦Â This approach doesnâ€™t install anything on your system and operates nicely as a test environment to let you get started with your first notebook. Once you navigate there, you will see a simple page with a few tabs, icons, and items.