What You Will Need
- ArcView 3.*x*
- Python Language
- AVPython extension

Using Python With **ArcView 3.*x***

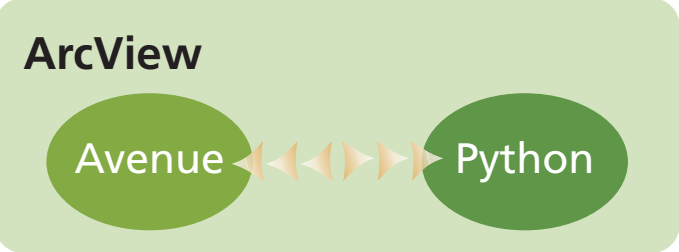
By Patrick Brown and Howard Butler, Iowa State University

After mastering the subtleties of Vtab and Ftab, many ArcView 3.*x* users realize that Avenue alone may not be able to meet all their challenges. Adding Python to the programmer's toolbox can give ArcView access to the Internet and to virtually any COM object with a published IDispatch scriptable interface. *[An IDispatch interface exposes objects, methods, and properties to programming tools and applications that support Automation.]* It can also enable ArcView to perform functions that just aren't that easy to do with Avenue. This article provides a few examples that demonstrate the benefits of using Python and ArcView together.

Introducing Python

Python is a portable, general purpose, interpreted, object-oriented programming language that works well with other programs or components. Best of all Python is free and, in some respects, resembles Avenue. With Python, users can create custom modules, classes, methods, or functions. Python also allows programmers to "stand on the shoulders of others" by using modules that provide a wide range of functionality such as database access, imaging libraries, network protocol usage, and XML parsing. For a programmer, this means that a considerable amount of work has already been done.

Developed by Guido van Rossum, Python has been evolving for more than 10 years. Some of Python's features were derived from a language called ABC, which was designed specifically to teach programming concepts to nonprogrammers. Python also includes features commonly associated with other programming languages such as object-oriented classes, exception handling, and dynamic data types. These features, combined with its internal consistency, readability, and the fact that it "fits in your brain," make Python a powerful language.



AVPython allows the Python language to be embedded in ArcView. This integration is bidirectional.

Why Learn Python?

In June 2002, the authors of this article developed an ArcView extension called avTerra that streamlines the process of getting TerraServer imagery into ArcView by allowing ArcView to act as a client to the Microsoft TerraServer instead of downloading files through a Web browser. It provides nationwide digital orthophoto quarter quadrangle (DOQQ) and digital raster graphic (DRG) coverage at six resolution levels. avTerra utilizes a Web Service (i.e., Simple Object Access Protocol or SOAP) interface that TerraServer provides for programmers. It was developed with Python and integrated into ArcView using AVPython. *[AVPython is an ArcView extension that embeds the Python programming language within ArcView 3.x. The avTerra application received first place in the Best Software Integration category of the Map Gallery at the 2002 ESRI International User Conference.]*

Although avTerra could have been developed using Java, C/C++, or even Visual Basic, Python had some features that made it very attractive. AVPython closely integrates Avenue with Python, a high-level scripting language that allows a developer to call a Python module directly from ArcView and loop back to call an ArcView script from Python. Python also fulfilled other development requirements such as SOAP support, integration with COM, XML processing capabilities, and a full-featured imaging library. Python also has a thriving user community that encourages participation and continually adds new capabilities to the language. Finally, it is a very developer-friendly language that does most of the heavy lifting and does not require the user to "reinvent the wheel" every time a function is needed.

Python's use of indentation to define blocks within classes, functions, and various flow-control statements makes it easy to read. In essence, the Python interpreter notices the logical structure of the program. Consecutive lines that are indented in the same way are part of the same block. Indentation makes revisiting code written a year ago in a language that might only be used occasionally much easier.

How Python Works With ArcView

Python code is called from ArcView by loading AVPython, the Python language support extension, into a project. This extension was developed by Bruce Dodson of ESRI Canada and has been released under an open-source licensing agreement. The extension includes a number of scripts that can be called from an ArcView project in order to interface with Python. Passing expressions to Python from an ArcView script is fairly easy and requires nothing more than calling the appropriate Avenue script and passing it a string. The string can be used to import a Python module, call a Python function, or execute Python code.

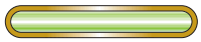
This article focuses primarily on the Python.Exec and Python.Eval scripts. Python.Exec executes Python code in a private scope. Any local variables created by this code will be discarded when Python.Exec is next called. It returns 0 when successful and -1 on error. Python.Eval evaluates a Python expression and returns a string. If an error occurs, nil is returned. Because it evaluates the Python expression in the context of the last Python.Exec call, all local variables from the previous call are available.

Using Python With ArcView

Using Python with ArcView requires copies of both Python and the AVPython extension. Both can be obtained on the Web.

1. To download and install Python, go to the official Python site (www.python.org) and download Python 2.2, or go to ActiveState (www.activestate.com) and download ActivePython, a free commercial build of the Python core library plus a suite of Windows tools including the Pythonwin integrated development environment (IDE), support for Python ASP, and the PythonCOM system. *[Pythonwin was written by Mark Hammond and can be downloaded separately from starship.python.net.]*

User Level



2. Visit avpython.sourceforge.net to download the AVPython extension. Be sure to read the brief but useful documentation that accompanies this extension.

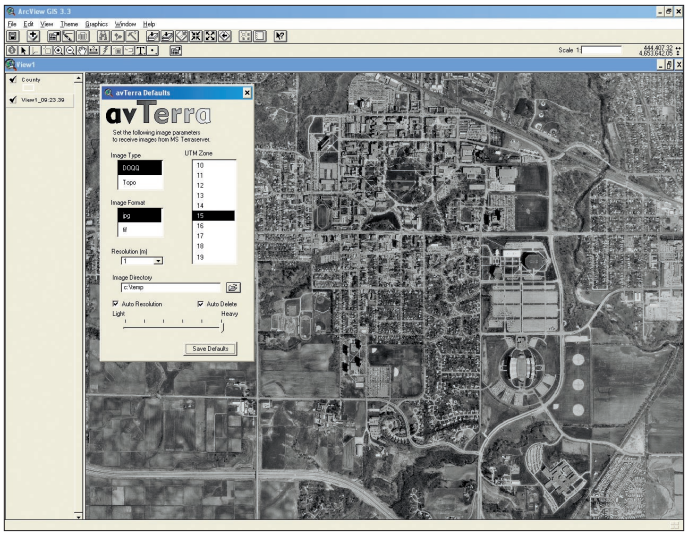
Example 1—Calling a Python Module

Python modules are called by first importing a module. In Python parlance, a module is analogous to an ArcView extension and a function is analogous to a script. Import the module by running the Python.Exec script and passing it the name of the Python module to be imported. In this case, the module is called random and is located in the Lib directory where Python was installed. Importing exposes all of the methods and objects available in the random module. To import the random module so that the methods or objects can be called from ArcView, create a new script in ArcView and add the following line:

```
av.Run("Python.Exec", "import random")
```

All methods and objects within the random module are now accessible. Next, a Python method will be called using the Python.Eval script. This script evaluates the expression within the context of the last Python.Exec call. After Python evaluates the expression, a string is returned. Add the lines of code shown below to the ArcView script previously created. This code calls the randint (random integer) function. Compile and run the script. The random integer function requires a minimum range value (1) and a maximum range value (100). It will return an integer within the specified range, which can be displayed using a message box.

```
theResult = av.Run
   ("Python.Eval","random.randint(1,100)")
msgbox.info
   (theResult.AsString,"The random number is:")
```



avTerra allows users to directly import TerraServer images into ArcView.

```
Listing 1
'randomSelect.ave
theView = av.GetActiveDoc
theTheme = msgbox.ListAsString(theView.GetThemes
"Choose a Theme"," Theme")
theFtab = theTheme.GetFtab

theNumRecs = theFtab.GetNumRecords

'index range to pass to the randint function
theMin = 0
theMax = theNumRecs - 1

'Evaluates the Python script:
'random.randint(minimum value, maximum value)
'passing the function the theMin and theMax values.
Values need to be 'passed as strings but will be
evaluated as integers by Python
av.Run("Python.Exec", "import random")

theEvalStr ="random.randint("+theMin.AsString+","+the
Max.AsString+")"
theResult = av.Run("Python.Eval",theEvalStr)

'make the selection on the map using theResult
'which is returned from the Python script
theBitMap = Bitmap.make(theNumRecs)
theBitMap.Set(theResult.AsNumber)
theFtab.SetSelection(theBitMap)
theFtab.UpdateSelection
```

Example 2—Making a Random Selection

Create a new script in ArcView, and copy and paste the code in Listing 1. This script will again import the random module and select the record in a shapefile associated with the returned random number.

Example 3—Calling ArcView From Python

This example demonstrates that scripts written in Python can be accessed from ArcView and vice versa. ArcView can be accessed from Python by importing a module called arcview. Currently the arcview module has one supported function called avexec. This function runs the Avenue code and returns the result as a string and can be used as a convenient way to access the ArcView object model even if the bulk of the program is written in Python. This method can also be used to access global variables defined in Avenue.

For this example, open a text editor or the Python IDE and create a new Python script that contains the code shown in Listing 2. **Remember that indentation matters in Python.** Comment statements that begin with a # may be excluded. Save the file as avMessage.py

Continued on page 38

Using Python With **ArcView 3.*x***

Continued from page 37

```
Listing 2
#avMessage.py

def infoBox(theString):        #this function expects a string

   from arcview import avexec  #regular Python scripts would import
                               #modules at the beginning of the script

   avexec('msgbox.info("' + theString + '","Message from Python")')

                               #calls a message box from AV and inserts
                               #theString passed from the Avenue script
```

```
Listing 3
'script2.ave

'string to pass to the Python script
theString = "This message box was called from Python"

'import the avMessage module you created
av.Run("Python.Exec", "import avMessage")

'execute the infoBox() function you created in the avMessage module
'the function expects a string.
av.Run("Python.Eval", "avMessage.infoBox('" + theString + "')")
```

in the Python/Lib directory or in the same directory as the ArcView project. avMessage is a simple Python module. Within this module is a function called infoBox, which requires a string. Notice the call to the avexec function. This function opens a message box and uses the string passed from ArcView. The Avenue script that invokes Python is shown in Listing 3. It calls the Python script, avMessage.py, and passes the string to the infoBox function. Then avMessage.py opens up a message box in ArcView and prints the message contained in the string.

Exploring Other Examples

This introduction should give users a good foundation for designing ArcView applications that integrate Python. There are a number of other examples worth exploring. Each shows off some unique capabilities available when Python is used within ArcView. All these examples can be accessed from the Web sites listed in the Resources section at the end of this article.

Excel Example

The AVPython package includes an example that demonstrates how to push data into an Excel table using Python and COM rather than using Dynamic Data Exchange (DDE).

avTerra

The avTerra extension was developed to access imagery available on the Microsoft TerraServer site. Images are tiled and brought directly into ArcView. All of this is done using Python modules that support XML and COM. This extension also uses the Python Imaging Library, the Microsoft SOAP Open Source Toolkit, and Microsoft TerraServer's SOAP API. Fortunately, all these details are hidden from the user. Digging into the Avenue and Python code used in creating this extension illustrates the power and usefulness of Python.

MySQL Connector

For users of MySQL or other open source databases, the MySQL Connector demonstrates a non-ODBC method of querying and inserting data into a MySQL database from ArcView.

XML and HTML Scrape

This demonstration allows a user to click on a map of the United States to access United States Geological Survey (USGS) real-time stream gauge information. Python obtains the USGS stream gauge locations from an XML file generated by the Web site. The real-time stream flow information is parsed from an HTML table. Location and flow information are then added to the ArcView project, joined, and added to the View as an event theme.

Conclusion

ArcView and Python can be a powerful combination. Python provides an Avenue programmer with a method for scripting new functionality without resorting to lower level languages such as C/C++. Python's syntax is clean and feels familiar to an experienced Avenue programmer. It has both cross platform capabilities and a thriving user community. Digging through the examples in this article will provide ideas for integrating Python and ArcView.

Resources

Article Web site	www.gis.iastate.edu/python
Python language	www.python.org or www.activestate.com/Products/ActivePython
AVPython mail list	groups.yahoo.com/group/avpython-dev/
AVPython extension	avpython.sourceforge.net
avTerra extension	hobu.stat.iastate.edu/avTerra

About the Authors

Patrick Brown is a GIS analyst at the Iowa State University GIS Support and Research Facility. **Howard Butler** is a GIS analyst working at the Iowa State University Statistical Laboratory.

The authors wish to acknowledge Bruce Dodson of ESRI Canada for his suggestions concerning this article and the continued development of AVPython.