PRACTICAL LECTURE
Python and Proprietary GIS

Fredrik Lindberg

Urban Climate Group
Department of Earth Sciences
University of Gothenburg

Python and Proprietary GIS

Most GIS and remote sensing software systems now have bindings to Python

To combine Python GIS scripting between different software systems is a challenge

Example of desktop software systems available for Python scripting:

- ESRI ArcGIS Pro
- ESRI ArcGIS Desktop 10.8 (Supported until 2026, no 10.9 planned)
- Safe FME Desktop
- ENVI/IDL

Python and Safe FME

Not tested by teacher but the use of Python transformers (e.g. PythonCaller) are available.

FME Python API Documentation:

http://docs.safe.com/fme/html/fmepython/index.html

Tutorial:

https://community.safe.com/s/article/python-and-fme-basics

PythonCaller:

https://docs.safe.com/fme/html/FME_Desktop_Documentation/FME_Transformers/Transformers/pythoncaller.htm

Python and ArcGIS Pro

As with **osgeo**-related products you must point to the correct Python installation linked to ArcGIS on your computer.

Accessing Python for ArcGIS Pro:

1. Use the installed IDE or Notebook from the Start menu (Windows)
2. Access via VSCode similar as with osgeo-products (live example later)

Investigate what Python that are used and what environment path settings that are used by exploiting the built in sys module:

```
Import sys
sys.executable
sys.path
```

Python and ArcGIS

Tool in ArcGIS is available via the **arcpy** Python module

Running a tool:

```
import arcpy
arcpy.analysis.Buffer("c:/temp/dronephotos.shp", "c:/
temp/dronephotos_buffer500.shp ", "500 METERS")
```

Getting results from a tool:

When a geoprocessing tool is executed, the results of the tool are returned in a Result object. Typically, this object is the path to the output dataset produced or updated by the tool. In other cases, it may contain other value types.

```
result = arcpy. analysis.Buffer("rivers", "riverBuf", "50 METERS")
print result
C:\Portland\Portland_OR.gdb\riverBuf
```

Python and ArcGIS

Using environment settings:

Geoprocessing environment settings can be thought of as additional parameters that affect a tool's results. They differ from normal tool parameters in that they are set separately from the tool and are interrogated and used by tools when they are run.

```
arcpy.env.workspace = "c:/data/Portland.gdb“
```

Using functions:

ArcPy exposes a number of functions to better support geoprocessing workflows. Functions can be used to list certain datasets, retrieve a dataset's properties, check for existence of data, validate a table name before adding it to a geodatabase, or perform many other useful scripting tasks.

```
print(arcpy.Exists("c:/data/Portland.gdb/streets"))
False
```

Python and ArcGIS

Using classes:

ArcPy classes, such as the SpatialReference and Extent classes, are often used as shortcuts to complete geoprocessing tool parameters that would otherwise have a more complicated string equivalent.

```
spatial_ref = arcpy.SpatialReference("Hawaii Albers Equal Area Conic")
```

Working with modules:

ArcPy includes modules covering other areas of ArcGIS. ArcPy is supported by a series of modules, including a data access module (arcpy.da), a mapping module (arcpy.mapping), an ArcGIS Spatial Analyst module (arcpy.sa), and an ArcGIS Network Analyst module (arcpy.na).

For example, the tools of the arcpy.sa module use tools in the Spatial Analyst toolbox but are configured to support Map Algebra. Thus, executing arcpy.sa.Slope is the same as executing the Slope tool from the Spatial Analyst toolbox.

```
arcpy.sa.Slope("c:/data/Portland.gdb/streets")
```

arcpy – loop example

Copy a large number of shape files into a geodatabase

```
import os
import arcpy

# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"

# Use the ListFeatureClasses function to return a list of  shapefiles.
featureclasses = arcpy.ListFeatureClasses()

# Copy shapefiles to a file geodatabase
for fc in featureclasses:
   arcpy.CopyFeatures_management(fc, os.path.join("c:/base/output.gdb",
                                                  os.path.splitext(fc)[0]))
```

Another loop example to run the clipping tool over multiple shapefiles

https://www.youtube.com/watch?v=M1BPbGRS2JQ (ArcGIS 10.x)

VSCode and ArcGIS

- Start ArcGIS Python Command Prompt from the start menu in Windows
- Locate Code.exe and execute
- Start a new project and choose the interpreter based on your ArcGIS installation:
  - ArcMap 10.x – C:\Python27\ArcGIS10.3\python.exe
  - ArcGIS Pro – c:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe
- Ready to start coding
- Sometimes the ArcGIS 10.x python version have an issue with code completion using arcpy. This is solved by running one simple script using arcpy function, e.g.:

```
Import arcpy
grid = arcpy._RasterToNumPyArray('c:/temp/DSM_LondonCity_1m.tif‘)
```