GIS Tips & Tricks

By Chris Griesbach, Andrew Collins, and Al Karlin, Ph.D, CMS-L, GISP

New to Python? Here's how to get started.

Although some computer programming languages come and go in popularity (think BASIC from the 1980's and 1990's), others gain in popularity and spread to accommodate many different niches. The Python language has now gone through three major upgrades and numerous interim revisions. The current version, Python 3.7.4, was released in July 2019 and has been widely accepted across many computational disciplines, including physics and astronomy, mathematics, gaming and graphic modeling, and geographic information systems (GIS), and across many hardware platforms.

However, for some GIS practitioners, getting started with Python may be intimidating. There are thousands of pages of documentation, thousands of web-pages (see https://www.python.org), and likely more than 100 different "python editors," including on-line versions. So, where is a novice to start?

If you are working in the Esri or QGIS GIS environments, the answer is simple. Use ModelBuilder or Graphical Modeler in ArcGIS or QGIS respectively, to construct your workflow, and then export that model to Python. These are the simple steps you can take to build a custom Python script; (1) construct an Esri ModelBuilder (or QGIS Graphical Modeler) workflow, (2) export the model to a Python script, and (3) use any text editor to customize that Python script. The following illustrates the Esri workflow, but the workflow in QGIS is nearly identical.

Suppose you have two TIF rasters, East.tif and West.tif, that you would like to clip to a 1500 meter x 1500 meter grid and merge into a single raster for easier analysis. Knowing this simple set of parameters, you can build a workflow in ModelBuilder by adding two Clip operations—one for East.tif and one for West.tif, with a Clipping_Grid.shp shapefile used by each—and having the result of each operation input into the Mosaic to New Raster tool. The model might look like Figure 1.

Python 3.7.4, was released in July 2019 and has been widely accepted across many computational disciplines, including physics and astronomy, mathematics, gaming and graphic modeling, and geographic information systems (GIS), and across many hardware platforms.

This ModelBuilder model will clip the two rasters and mosaic them together into a single output TIFF file. But suppose you have a long list of rasters for which you want to accomplish the same task. There are ways to do this in ModelBuilder, but they are time-consuming or more complex than the alternative, which is to add just a couple of lines of simple code to the model you have just built.

Start by clicking "Model" in the upper left corner of the ModelBuilder window. From the menu, select "Export," then "To Python Script…" This action will save the script as a .py file to the directory of your choosing. You can open this

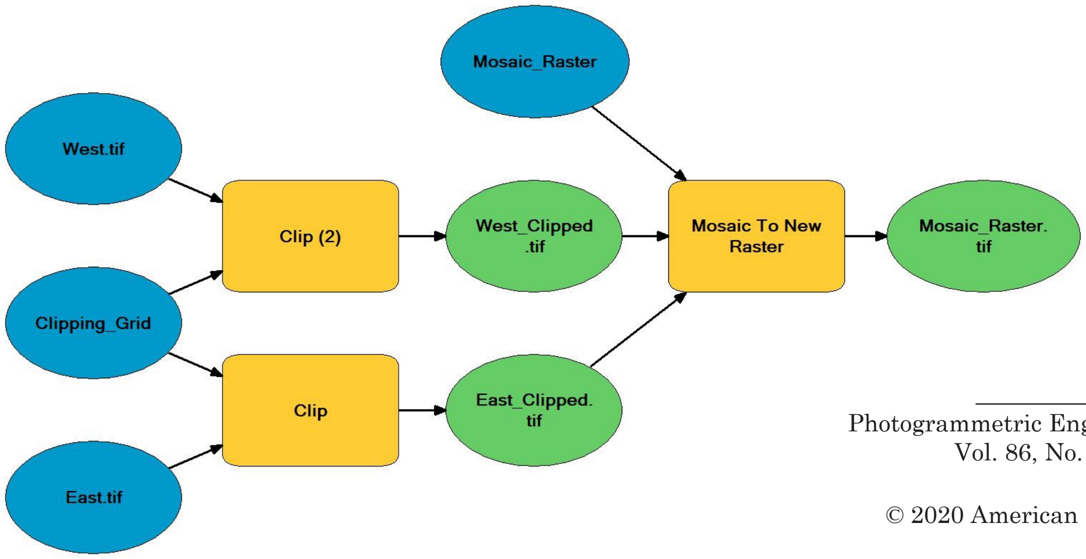

Figure 1.

Photogrammetric Engineering & Remote Sensing
Vol. 86, No. 1, January 2020, pp. 11–12.
0099-1112/20/11–12

doi: 10.14358/PERS.86.1.11

file with a script editor, such as IDLE or with MicroSoft's ™ Notepad. The exported script for the model above looks like Figure 2.

```
# Import arcpy module
import arcpy


# Local variables:
East_tif = "East.tif"
Clipping_Grid = "Clipping_Grid"
East_Clipped_tif = "C:\\Script_Test\\Clipped_Rasters\\East_Clipped.tif"
Mosaic_Raster_tif = East_Clipped_tif
West_tif = "West.tif"
West_Clipped_tif = "C:\\Script_Test\\Clipped_Rasters\\West_Clipped.tif"
Mosaic_Raster = "C:\\Script_Test\\Mosaic_Raster"

# Process: Clip
arcpy.Clip_management(East_tif, "604500 3550500 700500 3745500", East_Clipped_tif, Clipping_Grid, "256", "ClippingGeometry", "NO_MAINTAIN_EXTENT")

# Process: Clip (2)
arcpy.Clip_management(West_tif, "604500 3550500 700500 3745500", West_Clipped_tif, Clipping_Grid, "256", "ClippingGeometry", "NO_MAINTAIN_EXTENT")

# Process: Mosaic To New Raster
arcpy.MosaicToNewRaster_management("C:\\Script_Test\\Clipped_Rasters\\East_Clipped.tif;C:\\Script_Test\\Clipped_Rasters\\West_Clipped.tif",
                                   Mosaic_Raster, "Mosaic_Raster.tif", "", "8_BIT_UNSIGNED", "", "1", "LAST", "FIRST")
```

Figure 2.

When the script is exported from ModelBuilder, it has simple comments (denoted by a "#") indicating the list of variables (inputs and outputs) and where in the processing workflow, each GIS tool is called (=used). This information is a great guide for seeing how the variables are defined and the order in which Python reads and executes operations. In this example, all the variables are hard-coded to the specific files and file paths used as inputs and outputs by the ModelBuilder model. In other words, the variables are not variable.

To make the model more flexible, you can start by defining a file path (instead of a singular file) where the input rasters are located. Setting this as the workspace (an environment variable) also allows you to perform any analytical, array, or list operations on the files in that folder without setting additional file path variables. You can then define your local variables, which in this case are the clipped rasters and merged raster output folders, clipping grid, and a list of the rasters in the input folder. Remember that it is always a good idea to leave informative comments in your code so that you or others can come back to it and know what each step is doing.

The new list of variables, with comments for reference, is displayed in Figure 3.

```
# Import arcpy module
import arcpy

# Copy and paste the file path to the folder containing the rasters that you wish to clip.
arcpy.env.workspace = r"C:\\Script_Test\\Original_Rasters"

# Local variables:
# Copy and paste the file path to the folder where you want the clipped rasters to go.
out = r"C:\\Script_Test\\Clipped_Rasters"

# Copy and paste the file path to the extent shapefile that you want the rasters clipped by.
Clipping_Grid = r"C:\\Script_Test\\Tile_Grid\\Clipping_Grid.shp"

# Copy and paste the file path to the merged raster output.
mosaic_out = r"C:\Script_Test\Mosaic_Raster"

# Make a variable that lists all of the rasters within your current workspace, defined above.
rasters = arcpy.ListDatasets("*", "Raster")
```

Figure 3.

From here, you can use the same Clip process as in the original exported Python script. However, instead of setting it up to run on specific files, you can wrap it into a simple "for" loop, which clips each raster in the list of rasters created earlier, Figure 4.

```
# Process: Clip
# Iterates through and clip each of rasters in the "rasters" list variable. This step also adds "Clipped_" to the beggining
# of each clipped raster file name and prints a message after each raster is successfully clipped.
for r in rasters:
    arcpy.Clip_management(r, "",out+ "\\" + "Clipped_" + str(r), Clipping_Grid, "256", "ClippingGeometry", "NO_MAINTAIN_EXTENT")
    print(r + " successfully clipped.")
```

Figure 4.

You can now execute the Mosaic to New Raster function using the same techniques already mentioned. To ensure you are manipulating the clipped rasters and not the originals, you can change the workspace environment variable to the folder location where the clipped rasters are stored. Then create a new list of the clipped rasters. Because the Mosaic to New Raster tool is set up to handle multiple files, a "for" loop is not required for this function. Instead, you can simply set the input as the list variable you just created and run the script, Figure 5.

```
# Process: Mosaic To New Raster
# Set the workspace to the location of the clipped rasters, which will become the input (as a list)
# for the "Mosaic to New Raster" tool.
arcpy.env.workspace = out

clipped_rasters = arcpy.ListDatasets("*", "Raster")

arcpy.MosaicToNewRaster_management(clipped_rasters, mosaic_out, "Mosaic_Raster.tif", "", "8_BIT_UNSIGNED", "", "1", "LAST", "FIRST")
```

Figure 5.

This is just the beginning of how you can get creative with Python and ModelBuilder and/or Graphical Modeler. You can tweak parameters for an individual tool or process, add error checks or multiple iterations, print statements to show you the progress of the script, define styles, integrate tools from other geospatial software packages, and so much more.

If you would like a copy of the final Python script or any other additional instructions, please contact us at geospatial@dewberry.com.

***Chris Griesbach**, **Andrew Collins**, and **Al Karlin, Ph.D., CMS-L, GISP** are with Dewberry's geospatial and technology services group in Tampa, Florida, and Denver, Colorado. Chris manages many aspects of lidar production and imagery analysis for Dewberry. Andrew is part of the quality management team, ensuring data integrity and developing streamlined procedures for all stages of elevation data production and delivery. As a senior GIS professional, Al works with all aspects of lidar, remote sensing, photogrammetry, and GIS-related projects.*