"You are not allowed to use Geoprocessing Results... ": The Students get smarter

This is part two of a post inspired by the thread in GeoNet entitled **I'm a student and I need a python script that i can use for ArcMap** and Modelbuilder was not permitted to generate the script (https://geonet.esri.com/thread/116880). In my previous blog, You are not allowed to use Modelbuilder": When Instructors need to get smarter [1], I discussed how to use the Geoprocessing results window to generate a script stub by using a process that works manually to generate it. Last time, we were left with this somewhat horrendous, but functional script... stub_01.py which I won't even attempt to format to make it readable.

```
'''
stub_01.py
'''
# Replace a layer/table view name with a path to a dataset (which can be a layer file) or create the
layer/table view within the script
# The following inputs are layers or table views: "AOI_mtm9"
arcpy.MakeFeatureLayer_management(in_features="C:/!test/mapping/shapefiles/AOI_mtm9.shp",out_layer="AOI_out",wh
ere_clause="#",workspace="#",field_info="FID FID VISIBLE NONE;Shape Shape VISIBLE NONE;Id Id VISIBLE NONE")
arcpy.MakeFeatureLayer_management(in_features="C:/!test/mapping/shapefiles/RandomPnts.shp",out_layer="Randout",
where_clause="#",workspace="#",field_info="FID FID VISIBLE NONE;Shape Shape VISIBLE NONE;Id Id VISIBLE NONE;X X
VISIBLE NONE;Y Y VISIBLE NONE")
# Replace a layer/table view name with a path to a dataset (which can be a layer file) or create the
layer/table view within the script
# The following inputs are layers or table views: "Randout", "AOI_out"
arcpy.SelectLayerByLocation_management(in_layer="Randout",overlap_type="INTERSECT",select_features="AOI_out",se
arch_distance="#",selection_type="NEW_SELECTION")
# Replace a layer/table view name with a path to a dataset (which can be a layer file) or create the
layer/table view within the script
# The following inputs are layers or table views: "Randout"
arcpy.CopyFeatures_management(in_features="Randout",out_feature_class="C:/!test/mapping/shapefiles/Result.shp",
config_keyword="#",spatial_grid_1="0",spatial_grid_2="0",spatial_grid_3="0")
```

This stub will now be dissected and used to generate a script for use in a toolbox. The assumptions...or the new requirements are (clever Instructor)

- ❏ Start ArcMap with a new blank project.
- ❏ I have switched up the files to use because of former students passing on answers to this class
- ❏ The workflow you develop must add the input data and the results to ArcMap
- ❏ You still cannot use Modelbuilder

So the script so far, will do that but it is somewhat ugly and is hardcoded for a particular set of data. Of course you would provide documentation on how to edit your script so that the user could perform the workflow with their data, specifying the inputs and outputs.

Mark so far **C-** ...

---

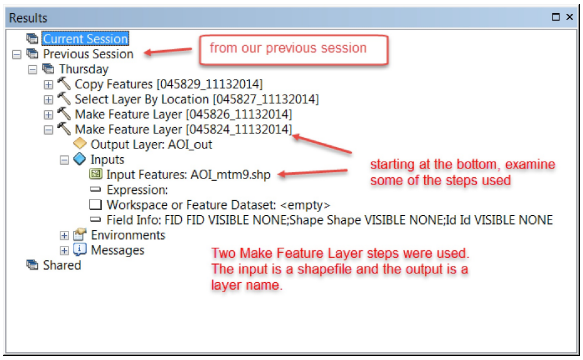
[1] Dan Patterson's Blog "You are not allowed to use modelbuilder": When Instructors Need to get smarter

Method 2 Expand the script...and see how far you can get

We will expand upon the workflow used previously. In that example, we used 2 feature layers to perform a select by location with the results copied to a new shapefile. This workflow will be taken to the next step.

1 Start with your last project, which you saved to disk, and open the Geoprocessing Results window.

2 Examine some of the steps used to create the script stub in more detail. In the previous session, the input to the Make Feature Layer tool was a shapefile residing on disk and no other parameters were specified. [2]
Feature layers are useful since they contain a reference to the data and the symbology that exists when the feature layer was last used. It is suggested that you save the feature layer to the same location with the same filename (.lyr extension) as the input file.



The syntax shown in the help files is:

MakeFeatureLayer_management (in_features, out_layer, {where_clause}, {workspace}, {field_info})

The only two required parameters are the features required to create the layer and the layer's name itself. The other parameters are optional (enclosed in curly brackets... { optional }. There is really no point in just replicating the tool exactly since it would be redundant. The clever student wants to simplify a workflow for the user and generate a tool that will be useful.

3 Begin modifying the script stub created in the last session and account for the new files to use. The comments have been temporarily removed and the **arcpy** module is imported as one of the first functional lines of the script. If it has been previously loaded, it won't be loaded again so it will have no detrimental impact on script run-time. To begin, translate this long-winded sequence...

```
'''
stub_02a.py
'''
import arcpy
arcpy.MakeFeatureLayer_management(in_features="C:/!test/Script_Demo/Shapefiles/Maija
_Poynts.shp",out_layer=" ....
... cont'd
```

to something more manageable by separating the required inputs and dumping any optional

[2] Make Feature Layer help

inputs that aren't necessary or desired and repeating it twice for each shapefile (stub_2a.py).

```
'''
stub_02a.py
'''
import arcpy

in_features="C:/!test/Script_Demo/Maija_Poynts.shp"
out_layer="Maija_out"
arcpy.MakeFeatureLayer_management(in_features,out_layer)

in_features="C:/!test/Script_Demo/Polly_Ghon.shp"
out_layer="Poly_out"
arcpy.MakeFeatureLayer_management(in_features,out_layer)
... cont'd
```

Hopefully, the clever student has been introduced to python objects and has been reading upon ArcToolbox tools and their parameters and has quickly realized that someone isn't going to want to perform the analysis on the exact same files you did, nor are they going to be inclined to edit your script to specify their desired inputs and outputs.

Now it is time to leave the **C-** group behind by making the script more generic and documenting it.

4 To make the script really useful, the script and its associated toolbox should allow the user to select multiple inputs. I also like to run a Python window to perform some of the steps in interactive mode prior to committing anything to script. Use any Python IDE that you want except the one in ArcMap since we want to do the testing independently of ArcMap at this stage.

From your readings, you will notice that many scripts use Python's **os** and **sys** modules to provide access to methods associated with your operating system and other system parameters. You can explore these by importing the modules and use Python's **dir** to get a listing of the options.

In interactive mode in a Python IDE, it may look something like this (greatly simplified):

```
>>> import arcpy
>>>
>>> import os
>>>
>>> dir(os)
[big snip .....'chdir', ... 'curdir', ... 'environ', ... 'getcwd', ... 'path', ...
'rename', ...]
>>>
>>> import sys
>>>
>>> dir(sys)
[big snip .....' ... ,'argv', ... ,'exit', ...]
>>>
```

At this point the variable names should be changed to reflect what they are. Several useful Python methods will also be used for handling file paths, names and extensions.

```
>>> .... Snip ...
>>> sel_from = "C:/!test/Script_Demo/Maija_Poynts.shp"   # the points to select
>>> os.path.split(sel_from)    # split the file name from the complete name
('C:/!test/Script_Demo', 'Maija_Poynts.shp')
>>> path, shp_name =  os.path.split(sel_from)  #returns a tuple, split into...
>>> path       # the path of the shapefile
'C:/!test/Script_Demo'
>>> shp_name   # the shapefile name
'Maija_Poynts.shp'
>>>
```

The above code snippet takes an input filename which will be the input to creating a feature layer and the select by attributes tools. The filename is split into its path and name components as can be seen from the outputs.

Some of you may have noticed that things to the left of an = sign are variable names and those to the right are its value. The **os** module contains a **path** module which contains a **split** function (lost yet ?). Keep these handy little snippets in a document so you can refer to them in the future when you have long forgotten but have a vague memory of doing something similar.

The **sys** module allows you to get script arguments (aka requirements) via the **argv** function. The scripts name is **sys.argv[0]**, and the remaining required variables are incremented accordingly. Exploiting this allows the script to evolve to accept parameters needed for the script to function and the user to choose. The new script stub, stub_02b.py, incorporates some of these ideas including the documentation and the new import statements.

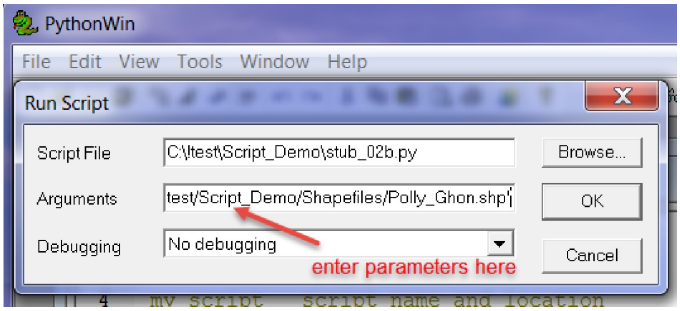
```
'''
stub_02b.py

my_script   script name and location
sel_from    The layer to select feature from
sel_by      The layer used to select the features in sel_from
parameters if run command line... a space delimited set of string values
"C:/!test/Script_Demo/Shapefiles/Maija_Poynts.shp"
"C:/!test/Script_Demo/Shapefiles/Polly_Ghon.shp"
'''
import arcpy
import os
import sys

# Get the input parameters
my_script = sys.argv[0]
sel_from = sys.argv[1]
sel_by = sys.argv[2]

# Make the feature layers and save to disk
path, shp_name =  os.path.split(sel_from)               # get the path and name
sel_from_lyr = shp_name.replace(".shp","")              # same name, different
extension
out_layer = path + "/" + sel_from_lyr + ".lyr"
out_layer.replace("\\","/")                             # Ooooo more reading
print "Creating layer : ", sel_from_lyr
arcpy.MakeFeatureLayer_management(sel_from, sel_from_lyr)
arcpy.SaveToLayerFile_management(sel_from_lyr, out_layer,"RELATIVE")
print "Created on disk: ", out_layer
..... BIG SNIP
# do it again
```

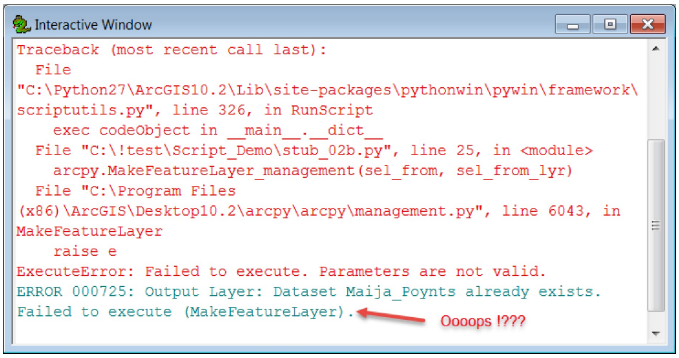
5 Run the script from within the your Python IDE. The syntax using PythonWin is shown as follows, with the required parameters entered in the arguments line.



Everything works fine the first time.

```
>>> Creating layer :  Maija_Poynts
Created on disk:  C:/!test/Script_Demo/Shapefiles/Maija_Poynts.lyr
```

The next time you run it to show your friends your genius, you get the error message shown to the right.



The problem can originate from many sources. For example, the variables weren't explicitly cleared in the script and upon closer examination, there are file locks on the input shapefiles.

Go back to the interactive window and use Python's dir function, the get a list of parameters still in memory.

```
>>> dir()
['__builtins__', '__doc__', '__name__', '__package__', 'arcpy', 'my_script', 'os',
'out_layer', 'path', 'pywin', 'sel_by', 'sel_from', 'sel_from_lyr', 'shp_name',
'sys']
>>>
```

The imported modules, arcpy, os and sys, are still present in memory as are all the variables created by the script. These should be cleared out from memory. The simplest is to simply close the PythonWin IDE. This will remove the file locks and the variable and import information from memory.

6 Time to step-up a grade. I am showing you the script that produced the layer files on disk, performed the select by location then exported the results to a new shapefile. Your task is to explain the lines that I haven't talked about.

```
'''
stub_02c.py

my_script   script name and location
sel_from    The layer to select feature from
sel_by      The layer used to select the features in sel_from

copy the line below to the script arguments to test
"C:/!test/Script_Demo/Shapefiles/ Maija_Poynts.shp" "C:/!test/Script_Demo/Shapefiles/Polly_Ghon.shp"

'''
import arcpy
import os
import sys

# Get the input parameters
num_args = len(sys.argv)
if num_args != 3:
  print "Read the script header for instructions"
  sys.exit()

arcpy.env.overwriteOutput = True    # overwrite existing files if possible

my_script = sys.argv[0]
sel_from = sys.argv[1]
sel_by = sys.argv[2]

# Make the feature layers and save to disk
path, shp_name =  os.path.split(sel_from)                  # get the path and name
sel_from_lyr = shp_name.replace(".shp","")                # same name, different
extension
out_layer = path + "/" + sel_from_lyr + ".lyr"
out_layer.replace("\\","/")                                # Ooooo more reading
print "Creating layer : ", sel_from_lyr
arcpy.MakeFeatureLayer_management(sel_from, sel_from_lyr) # create the output layer
arcpy.SaveToLayerFile_management(sel_from_lyr, out_layer,"RELATIVE")
print "Created on disk: ", out_layer

path, shp_name =  os.path.split(sel_by)
sel_by_lyr = shp_name.replace(".shp","")
out_layer = path + "/" + sel_by_lyr + ".lyr"
out_layer.replace("\\","/")
print "Creating layer : ", sel_by_lyr
arcpy.MakeFeatureLayer_management(sel_by, sel_by_lyr)
arcpy.SaveToLayerFile_management(sel_by_lyr, out_layer,"RELATIVE")
print "Created on disk: ", out_layer

arcpy.SelectLayerByLocation_management(sel_from_lyr,"WITHIN",sel_by_lyr)
result_file = path + "/Result.shp"
arcpy.CopyFeatures_management(sel_from_lyr,result_file)
arcpy.SelectLayerByAttribute_management(sel_from_lyr,"CLEAR_SELECTION") # clear any
selection
print "Creating Select by Location output...", result_file

# Clean up by deleting references to layers
arcpy.Delete_management(sel_from_lyr)
arcpy.Delete_management(sel_by_lyr)
arcpy.RefreshCatalog(path)
```