



Python Map Automation: Introduction to `arcpy.mapping` / `arcpy.mp`

Jeff Barrette

What is arcpy.mapping?

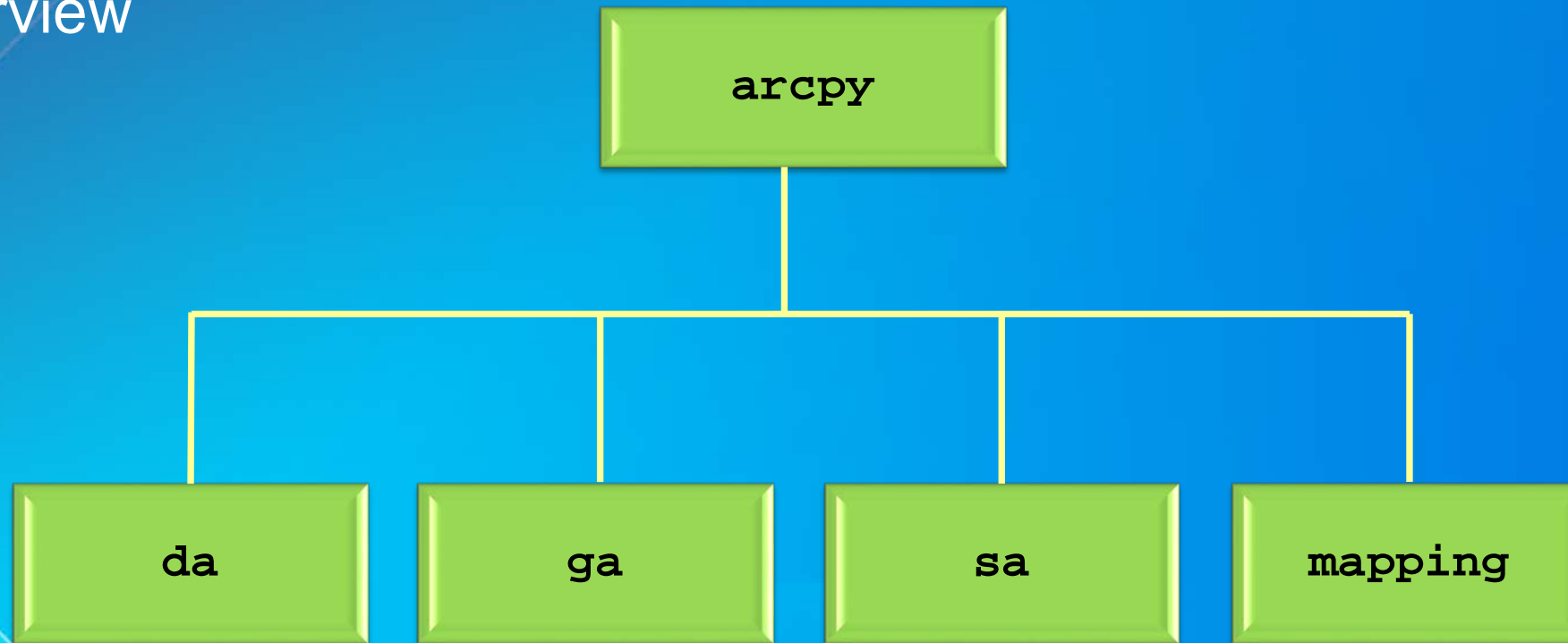
- Python mapping module that is part of the ArcPy site-package
- An API that allows users to:
 - manage map documents, layer files, and their contents
 - find a layer with data source X and replace with Y
 - update a layer's symbology in many MXDs
 - generate reports that lists document information
 - data sources, broken layers, spatial reference info, etc.
 - Automate the exporting and printing of map documents
 - Automate map production and create map books
 - extend Data Driven Pages capabilities

Who is arcpy.mapping for? Why was it built?

- An easy to use, productive scripting environment for the GIS Analyst
 - courser grained object model
 - not a complete replacement for ArcObjects
- An environment to use for basic map/layer management and map automation tasks
- A simple way to publish mapping tasks to the server environment
 - arcpy.mapping scripts can be easily published as geoprocessing tools

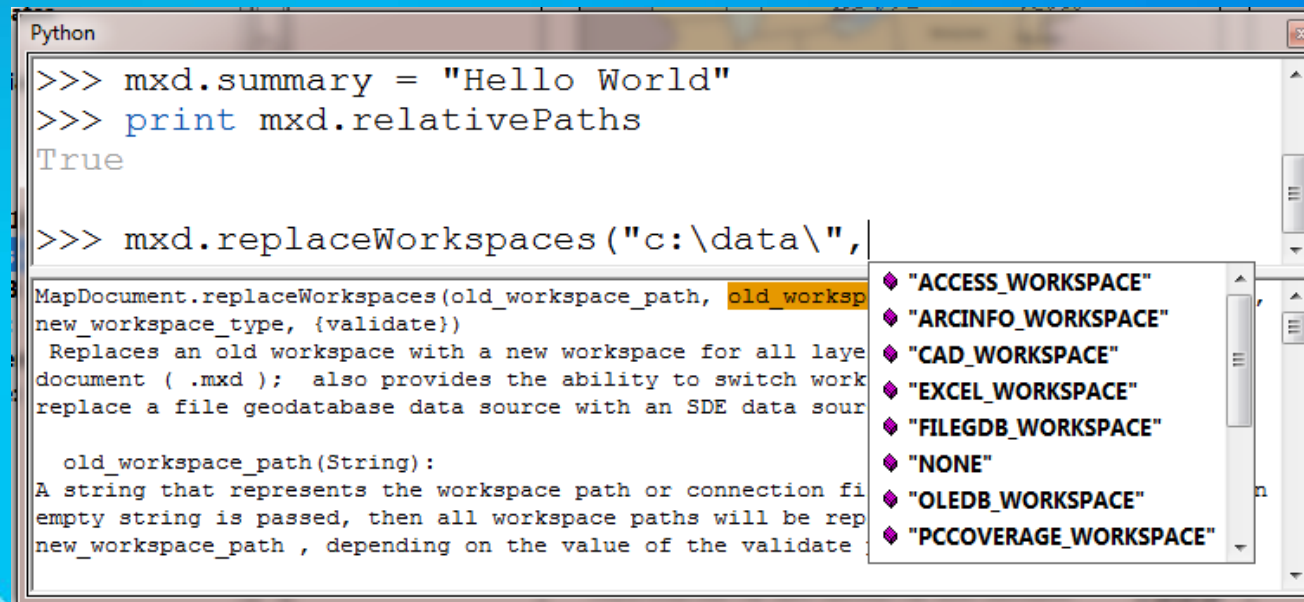
Tour of arcpy.mapping

Overview



Python window

- Quick and easy access to Python and arcpy
 - Gateway for new users to learn Python
 - Intellisense for all tools, methods and properties & help window
 - Quickly and efficiently execute tools



The screenshot shows a Python window with the following code and output:

```
>>> mxd.summary = "Hello World"
>>> print mxd.relativePaths
True

>>> mxd.replaceWorkspaces("c:\data\",
```

The dropdown menu lists the following workspace types:

- ◆ "ACCESS_WORKSPACE"
- ◆ "ARCINFO_WORKSPACE"
- ◆ "CAD_WORKSPACE"
- ◆ "EXCEL_WORKSPACE"
- ◆ "FILEGDB_WORKSPACE"
- ◆ "NONE"
- ◆ "OLEDB_WORKSPACE"
- ◆ "PCCOVERAGE_WORKSPACE"

Demonstration

- The Python Window and using the Desktop Help System

The image shows a screenshot of the ArcGIS Desktop interface. On the left, the Python Window is open, displaying the following code and its output:

```
>>> mxd.summary = "Hello World"
>>> print mxd.relativePaths
True

>>> mxd.replaceWorkspaces("c:\data\","
```

The Python Window also displays the documentation for the `replaceWorkspaces` method, which lists the following workspace types:

- ◆ "ACCESS_WORKSPACE"
- ◆ "ARCIINFO_WORKSPACE"
- ◆ "CAD_WORKSPACE"
- ◆ "EXCEL_WORKSPACE"
- ◆ "FILEGDB_WORKSPACE"
- ◆ "NONE"
- ◆ "OLEDB_WORKSPACE"
- ◆ "PCCOVERAGE_WORKSPACE"

On the right, the Desktop Help System is open, showing the "Code Sample" section. It contains two examples:

MapDocument example 1

The following script will create a separate MXD file for each data frame in a map document. The output map documents will be saved in data view mode so when each different map document is opened, the corresponding data frame will be the active data frame. The script will also set the title property of each output map document. Because this script uses a system path to the map document, it can be executed outside an ArcMap application. Note: Python strings cannot end with a backslash, even when the string is preceded by an r. You must use a double backslash. This becomes important when appending dynamic file names to a folder path.

```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\Project\Project.mxd")
for df in arcpy.mapping.ListDataFrames(mxd):
    mxd.activeView = df.name
    mxd.title = df.name
    mxd.saveCopy(r"C:\Project\Output\\" + df.name + ".mxd")
del mxd
```

MapDocument example 2

The following script demonstrates how the CURRENT keyword can be used within the ArcMap Python Interactive Window. This sample will update the first data frame's name and refresh the table of contents so the change will be immediately updated. Paste the following code into the Python Interactive window within a new ArcMap document.

```
import arcpy
mxd = arcpy.mapping.MapDocument("CURRENT")
arcpy.mapping.ListDataFrames(mxd)[0].name = "New Data Frame"
arcpy.RefreshTOC()
del mxd
```

Referencing map documents

MapDocument function

```
MapDocument (mxd_path)
```

MapDocument class

Methods

```
replaceWorkspaces
```

```
save
```

```
saveAsCopy
```

```
...
```

Properties:

```
activeDataFrame
```

```
author
```

```
credits
```

```
relativePaths
```

```
...
```


Referencing map documents (MXDs)

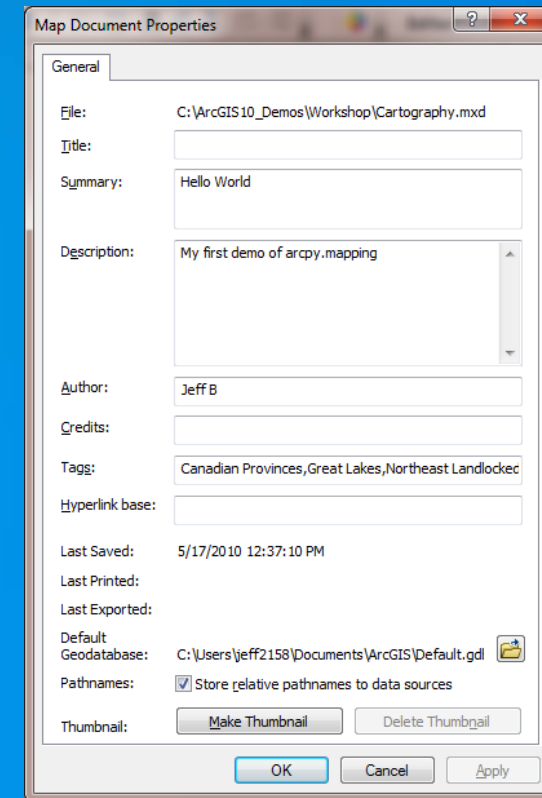
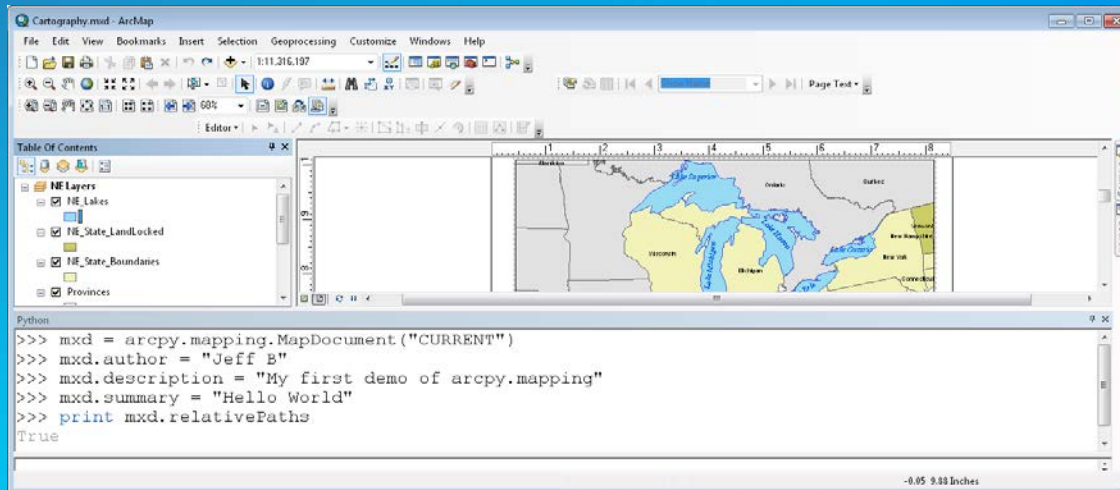
- Opening Map Documents (MXD) with `arcpy.mapping`
- Use the `MapDocument` function
 - Takes a path to MXD file on disk or special keyword "CURRENT"
- Reference map on disk
`mxd = arcpy.mapping.MapDocument(r"C:\some.mxd")`
- Get map from current ArcMap session
`mxd = arcpy.mapping.MapDocument("CURRENT")`

Referencing map documents (MXDs), cont.

- When using CURRENT
 - Always run in foreground (checkbox in script tool properties)
 - May need to refresh the application
`arcpy.RefreshActiveView()`
- Limitations and pre-authoring
 - No "New Map" function, so keep an empty MXD available
 - Can't create new objects (e.g., north arrow, data frame)

Demonstration

- Working with Map Documents (MXDs)



- Use Python Window to change map document property info
- Evaluate relative paths, last saved, etc.
- Change the active view
- Save changes out to a new file

Layers and data frames

- The “List” functions
 - ListLayers and ListDataFrames
 - Watch the list indexes (it is easy to forget to use [0])

```
df = arcpy.mapping.ListDataFrames(MXD)[0]
```
- Layer properties
 - Common properties are available (e.g., def query, visible)
 - All properties can be updated via layer (.lyr) files
- DataFrame properties and methods
 - Basic map navigation and settings

Layers and Data Frames

Layer functions

Layer
ListLayers
ListTableViews

AddLayer
AddLayerToGroup
InsertLayer
MoveLayer
RemoveLayer
UpdateLayer

...

Data Frame Class

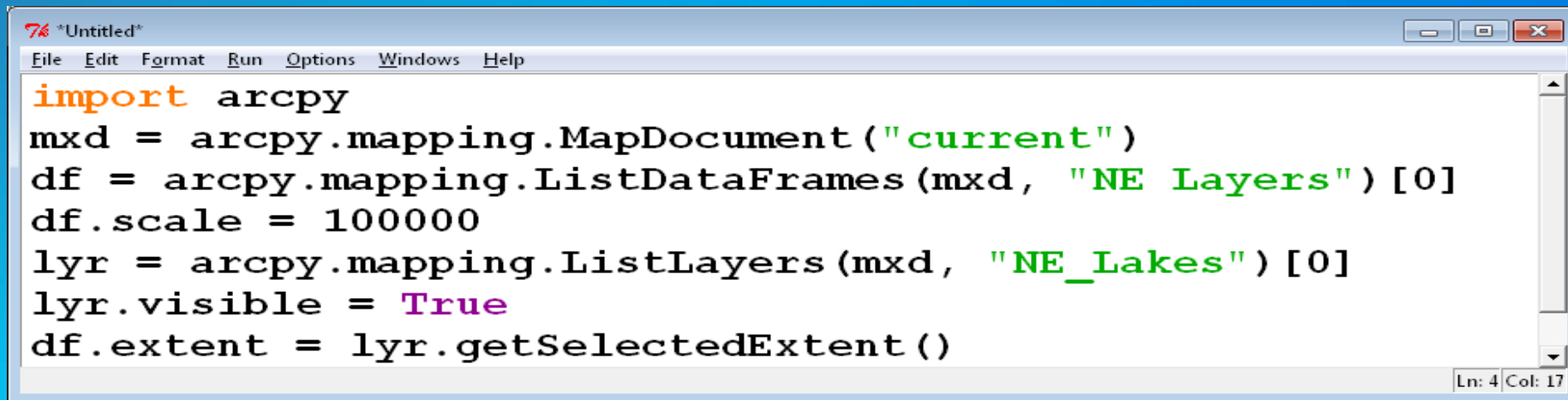
Methods
panToExtent(extent)
zoomToSelectedFeatures()

Properties:
credits
description
displayUnits
elementHeight
elementPositionX
extent
scale

...

Demonstration

- Working with Layers and Data Frames

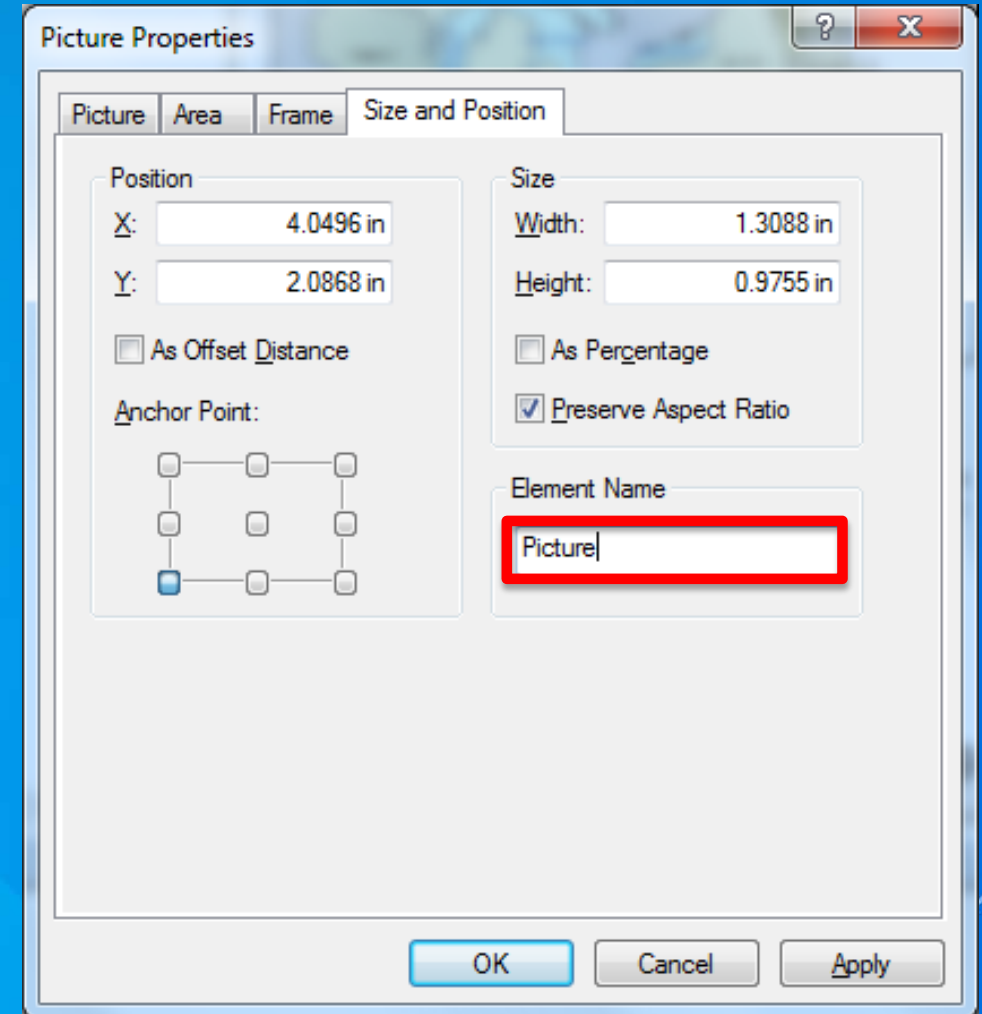


```
*Untitled*
File Edit Format Run Options Windows Help
import arcpy
mxd = arcpy.mapping.MapDocument("current")
df = arcpy.mapping.ListDataFrames(mxd, "NE Layers")[0]
df.scale = 100000
lyr = arcpy.mapping.ListLayers(mxd, "NE_Lakes")[0]
lyr.visible = True
df.extent = lyr.getSelectedExtent()
Ln: 4 Col: 17
```

- Find a layer and turns it on or off
- Modify the scale/rotation of a data frame
- Zoom to selected features

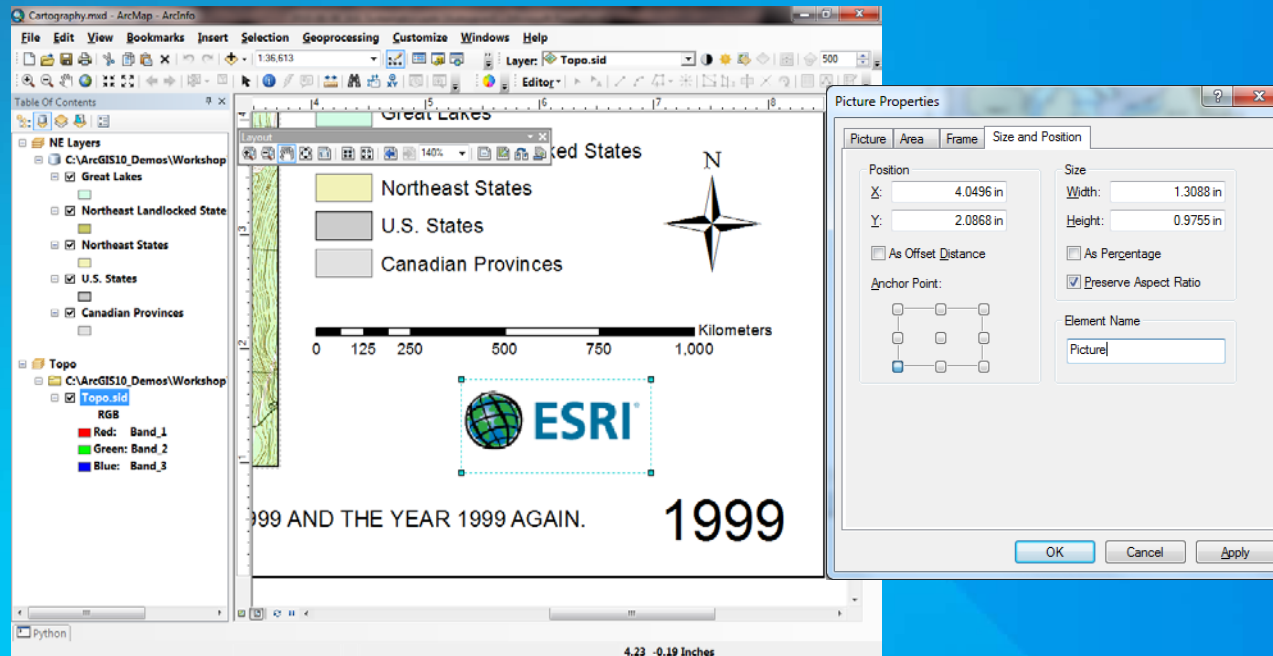
Page Layout Elements

- Must pre-author most elements
 - Uniquely name your layout elements
 - Set the appropriate anchor
- Only graphics and text can be cloned
- Move elements off the page



Demonstration

- Working with layout elements



- Find a picture element and change its data source

Printing, exporting, server publishing, map books

CLASSES

DataDrivenPages
PDFDocument

FUNCTIONS

ExportReport
ExportToAI
ExportToBMP
ExportToEPS
ExportToGIF
ExportToJPEG
ExportToPDF
ExportToPNG
ExportToSVG
ExportToTIFF
PDFDocumentCreate
PDFDocumentOpen
PrintMap
CreateMapSDDraft

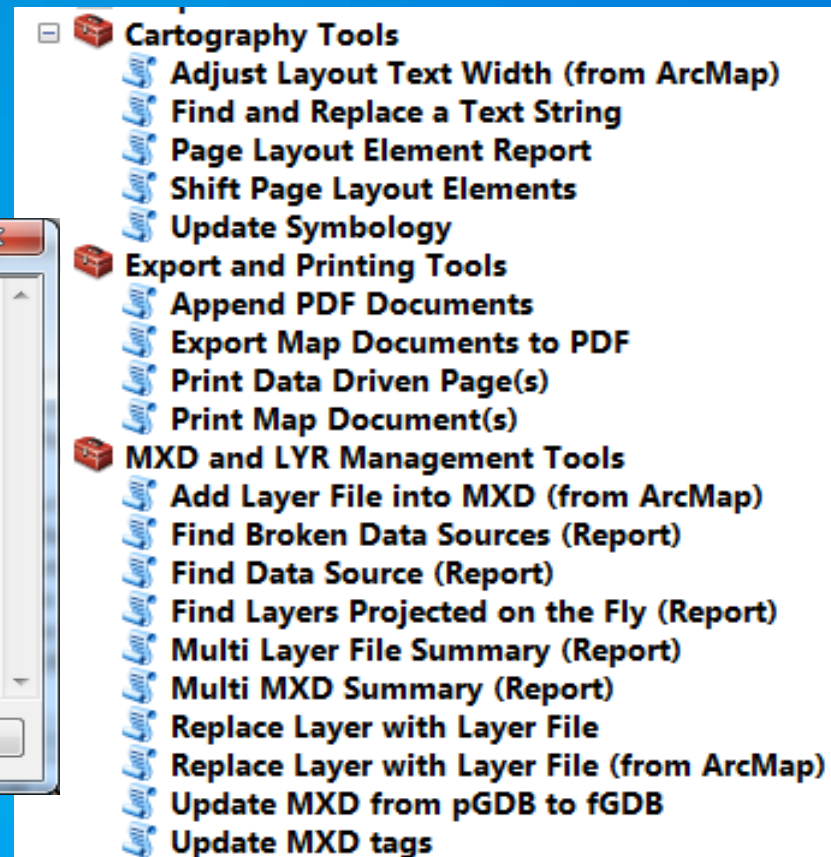
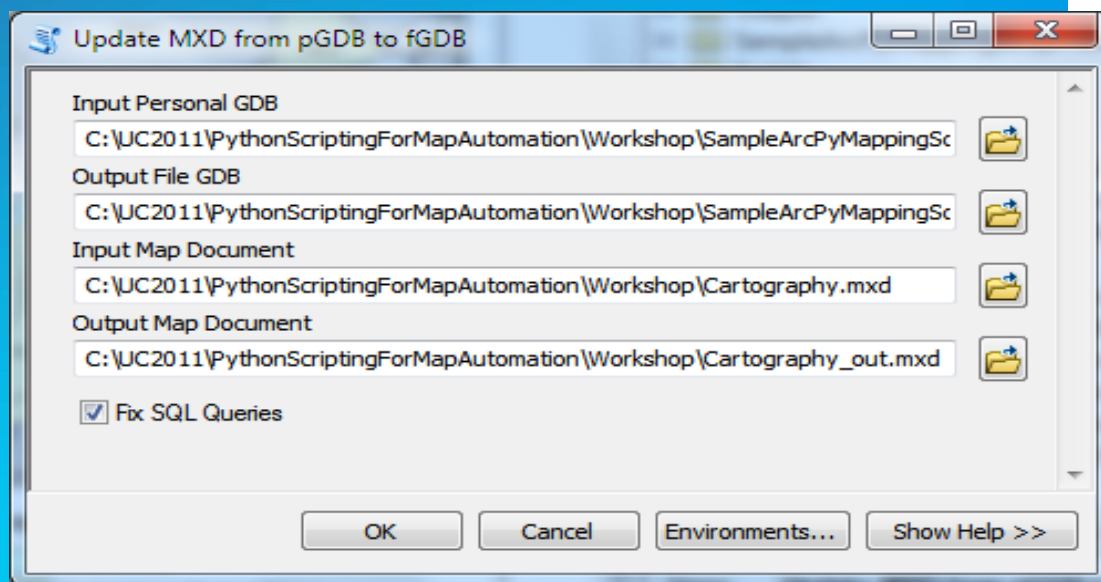
...

Updating data sources

- Use `arcpy.mapping` for migrating Map Documents and Layer files to new data sources
- Fancier scripts can help mitigate migration pain: SQL syntax changes, field name changes, etc
- A complete concept document is dedicated to this topic
 - “Updating and fixing data sources with `arcpy.mapping`”
 - <http://esriurl.com/8149>

Demonstration

- arcpy.mapping sample script tools



- Sample: <http://esriurl.com/4622>

Resources available

- ArcGIS Resource Center (web help) <http://esriurl.com/8148>
 - Alphabetical lists of classes and functions
 - Detailed discussions
 - Multiple sample scripts for each class and function topic
- ArcGIS Online – arcpy.mapping / Map Automation group
 - Download sample scripts <http://esriurl.com/8899>

Migrating to ArcGIS Pro

- Help Topic: Migrating arcpy.mapping from ArcMap to ArcGIS Pro
 - Python 3.4
 - ArcGIS project file (.aprx)
 - Stand-alone functions have moved to appropriate classes
 - `mapFrame.exportToPDF()`
 - `map.addLayer()`, `map.insertLayer()`, etc
 - Layer files have changed
 - DataFrame replace by Map, MapFrame, and Camera
 - New Layout object
 - Application always refreshes when using CURRENT

Related Sessions

Desktop Mapping: Building Map Books

- Wednesday, 10:15am – 11:30am, Room 2
- Thursday, 8:30am - 9:45am, Room 05 B

Advanced Map Automation with Python

- Wednesday, 3:15pm – 4:30pm, Room 15 B
- Thursday, 1:30pm - 2:45pm, Room 07 A/B



Understanding our world.