Python: Beyond the Basics

John Lam – Esri China (Hong Kong)
Richard Chau – Esri China (Hong Kong)
Basic rules

• Reference an MXD using a path or “current” keyword
  - When using CURRENT
    - Always run in foreground, may need to refresh (e.g., RefreshActiveView)

• Uniquely name all the objects (or at least the ones you want to find)

• Pre-author MXDs with all possible elements
  - Can’t create new objects (e.g., north arrow, data frames)
  - Author the extra elements off the page
  - No "New Map" function, so keep an empty MXD available

• This is not a replacement for ArcObjects – we are trying to draw a line in the sand

"The Line"
Demo: Multiple Element Layout Manager
Richard Chau

Python Map Automation - Beyond the Basics of arcpy.mapping
Cloning elements

• You can clone text and graphic elements
• This allows you to automate things like dynamic tables

```python
vertl = arcpy.mapping.ListLayoutElements(
    mxd, "GRAPHIC_ELEMENT", "VerticalLine")[0]
vertl.elementPositionX = xPos;
vertl.elementPositionY = 4
vertl.elementHeight = 3
for line in range(1, numColumns+1):
    vert_clone = vertLine.clone("_clone")
    xPos = xPos + colWidth
    vert_clone.elementPositionX = xPos
```

http://esriurl.com/5908

Python Map Automation - Beyond the Basics of arcpy.mapping
Demo: DDP with Dynamic Tables and Graphs

Richard Chau

Python Map Automation - Beyond the Basics of arcpy.mapping
arcpy.mapping group on ArcGIS Online

http://esriurl.com/5915

Python Map Automation - Beyond the Basics of arcpy.mapping
Performance tips

• Don’t keep calling list functions

```python
import map as arcpy.mapping
item1 = map.ListLayoutElements(mxd, wildcard="Item1")
item2 = map.ListLayoutElements(mxd, wildcard="Item2")
item3 = map.ListLayoutElements(mxd, wildcard="Item3")
```

• Call them once instead and iterate through the items

```python
for elm in arcpy.mapping.ListLayoutElements(mxd):
    if elm.name == "Item1": item1 = elm
    if elm.name == "Item2": item2 = elm
    if elm.name == "Item3": item3 = elm
```
Performance tips (continued)

- Use dictionaries

```python
dict = {}
for elm in arcpy.mapping.ListLayoutElements(mxd):
    dict[elm.name] = elm

dict["Item1"].text = "Dictionaries"
dict["Item2"].text = "are really"
dict["Item3"].text = "COOL!!!"
```
Functions for web map printing and server publishing

- **ConvertWebMapToMapDocument()**
  - Use with the ArcGIS web APIs for advanced web map printing workflows

- **CreateMapSDDraft()**
  - Automate publishing map documents to map services
Server printing out-of-the-box

- ArcGIS Server and the ArcGIS web APIs support web map printing via print services.
  - Out-of-the-box print service and template maps ship with Server
  - Print services sample: http://esriurl.com/6465

```python
app.printer = new esri.dijit.Print({
  "map": app.map,
  "templates": templates,
  url: app.printUrl,
}, dojo.byId("print_button"));
app.printer.startup();
```

```python
```
Advanced server printing with arcpy.mapping

- Build web apps with customized versions of the out-of-the-box print service

    Web application  arcpy.mapping  High-quality output (e.g. PDF)

- arcpy.mapping method for converting Web Maps to Map Documents:

    ConvertWebMapToMapDocument(webmap_json, {template_mxd}, {notes_gdb}, {extra_conversion_options})
Advanced server printing with arcpy.mapping

• Full capabilities of arcpy.mapping on the document
  - Swap out service layers for local vector data for vector PDF output
  - Export using advanced options
  - Export data driven pages
  - Export to PDF and insert additional pages (title page, reports, etc.)
  - Controlling the appearance of the legend
  - Etc.

• Return a printer-friendly output file (PDF, PNG, etc.)
• Online help and examples  http://esriurl.com/4600
Demo: Web app to export vector PDF using arcpy.mapping

Richard Chau
Web app to export vector PDF using arcpy.mapping

- Reference the custom arcpy.mapping based GP service

```javascript
var printUrl = "http://gilbert:6080/arcgis/rest/services/Austin/AustinPrint/GPServer/AustinPrint";

printTask = new esri.tasks.PrintTask(printUrl, {async: true});

params = new esri.tasks.PrintParameters();
params.map = app.map;
params.template.format = "PDF"
params.template = ptemplate;
printTask.execute(params, printComplete);
```
Web app to export vector PDF using arcpy.mapping

Basic vector web map printing: [http://esriurl.com/4601](http://esriurl.com/4601)

```python
# Reference the legend in the map document
legend = arcpy.mapping.ListLayoutElements(mxd, "LEGEND_ELEMENT")[0]

# get a list of service layers that are on in the legend because the incoming
# JSON can specify which service layers/sublayers are on/off in the legend
legendServiceLayerNames = [lslyr.name for lslyr in legend.listLegendItemLayers()]
if lslyr.isServiceLayer and not lslyr.isGroupLayer]
```
Advanced Server Printing: new function at 10.3

- **Layer.UpdateLayerFromJSON(json_layer_definition)**
  - Used in web map printing applications that support changing the renderer (or other properties) of dynamic web service layers.
  - Will apply the renderer (or other layer properties) as specified in the webmap_json to the corresponding vector layers staged in the template map document.

```python
# Convert the web map to a map document
result = arcpy.mapping.ConvertWebMapToMapDocument(Web_Map_as_JSON, templateMxd)
mxd = result.mapDocument

# Reference the data frame that contains the web map
df = arcpy.mapping.ListDataFrames(mxd, 'Webmap')[0]

# Reference the staged vector data that corresponds to the dynamic layer in the JSON
# This is the layer that will get updated based on the layer definition in the JSON
lyr = arcpy.mapping.ListLayers(mxd, 'U.S. States (Generalized)', df)[0]

# Read the JSON and extract the layer definition
# In this case we have hardcoded it to second operational layer
data = json.loads(Web_Map_as_JSON)
layerDefinition = data["operationalLayers"][1]["layerDefinition"]

# Update the staged vector layer with the layer definition (e.g. renderer info) from the JSON
lyr.updateLayerFromJSON(layerDefinition)
```

*Python Map Automation - Beyond the Basics of arcpy.mapping*
Publishing map services with arcpy.mapping

- `arcpy.mapping.CreateMapSDDraft(map_document, out_sddraft, service_name, {server_type}, {connection_file_path}, {copy_data_to_server}, {folder_name}, {summary}, {tags})`

- Workflow to convert map document to map service.

- Use Python scripts for:
  - Scheduled service updates. E.g. nightly.
  - Publishing automated analysis results.
  - Batch publishing.

Python Map Automation - Beyond the Basics of arcpy.mapping
Publishing map services with `arcpy.mapping`

Sample script: **CreateMapSDDraft**

```python
import arcpy

# define local variables
wrkspc = 'C:/Project/'
mmapDoc = arcpy.mapping.MapDocument(wrkspc + 'counties.mxd')
con = 'GIS Servers/arcgis on MyServer_6808 (publisher).ags'
service = 'Counties'
sddraft = wrkspc + service + '.sddraft'
sd = wrkspc + service + '.sd'
syntax = 'Population Density by County'
tags = 'county, counties, population, density, census'

# create service definition draft
arcpy.mapping.CreateMapSDDraft(mmapDoc, sddraft, service, 'ARCGIS_SERVER',
                                 con, True, None, summary, tags)

# analyze the service definition draft
analysis = arcpy.mapping.AnalyzeForSD(sddraft)

# stage and upload the service if the sddraft analysis did not contain errors
if analysis['errors'] == {}:
    # Execute StageService
    arcpy.StageService_server(sddraft, sd)
    # Execute UploadServiceDefinition
    arcpy.UploadServiceDefinition_server(sd, con)
else:
    # if the sddraft analysis contained errors, display them
    print analysis['errors']
```

Online help and samples: [http://esriurl.com/4598](http://esriurl.com/4598)
Publishing other service types with python

- **Create geoprocessing services**
  - `arcpy.CreateGPSDDraft()`

- **Create image services**
  - `arcpy.CreateImageSDDraft()`

- **Create geocoding services**
  - `arcpy.CreateGeocodeSDDraft()`
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
  - `DataFrames` replaced by `Map`, `MapFrame`, and `Camera`
  - New Layout object
  - Application always refreshes when using `CURRENT`
Spread the world #EsriAPUC