Python Map Automation – Beyond the Basics of arcpy.mapping

Jeff Barrette

Jeff Moulds

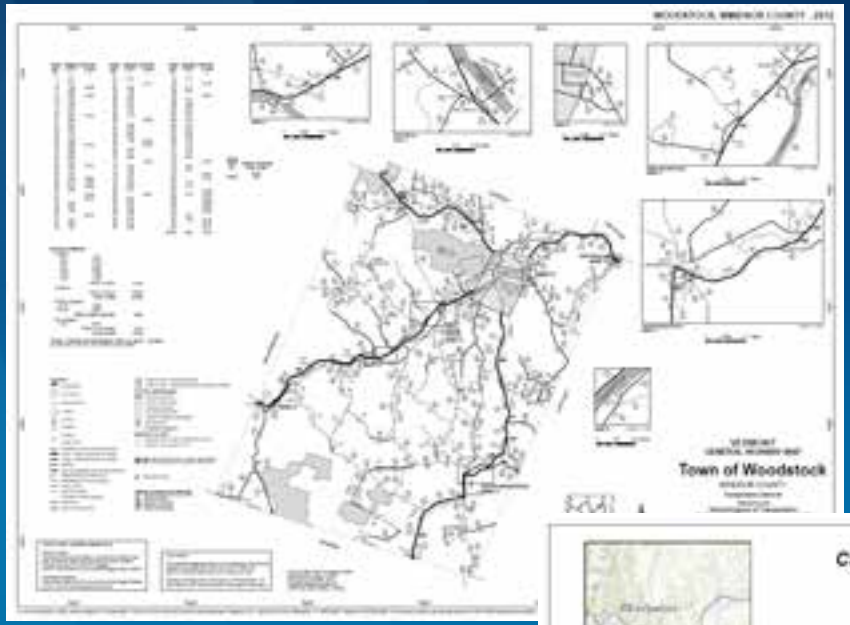
Technical Workshop

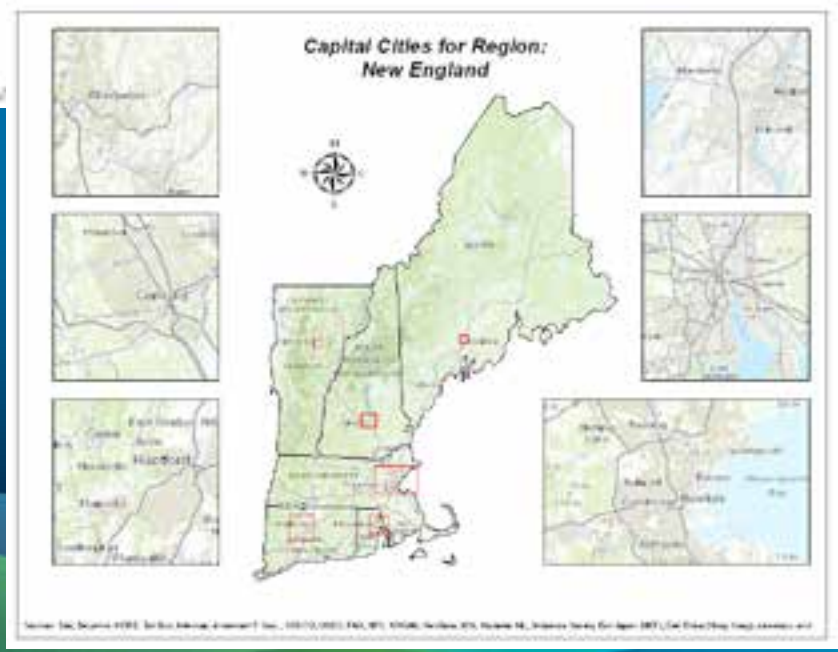
Basic rules

- 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

http://esriurl.com/5907





Sample Applications

Jeff Barrette

DEMO 1

DEMO 2

Cloning elements

http://esriurl.com/5908

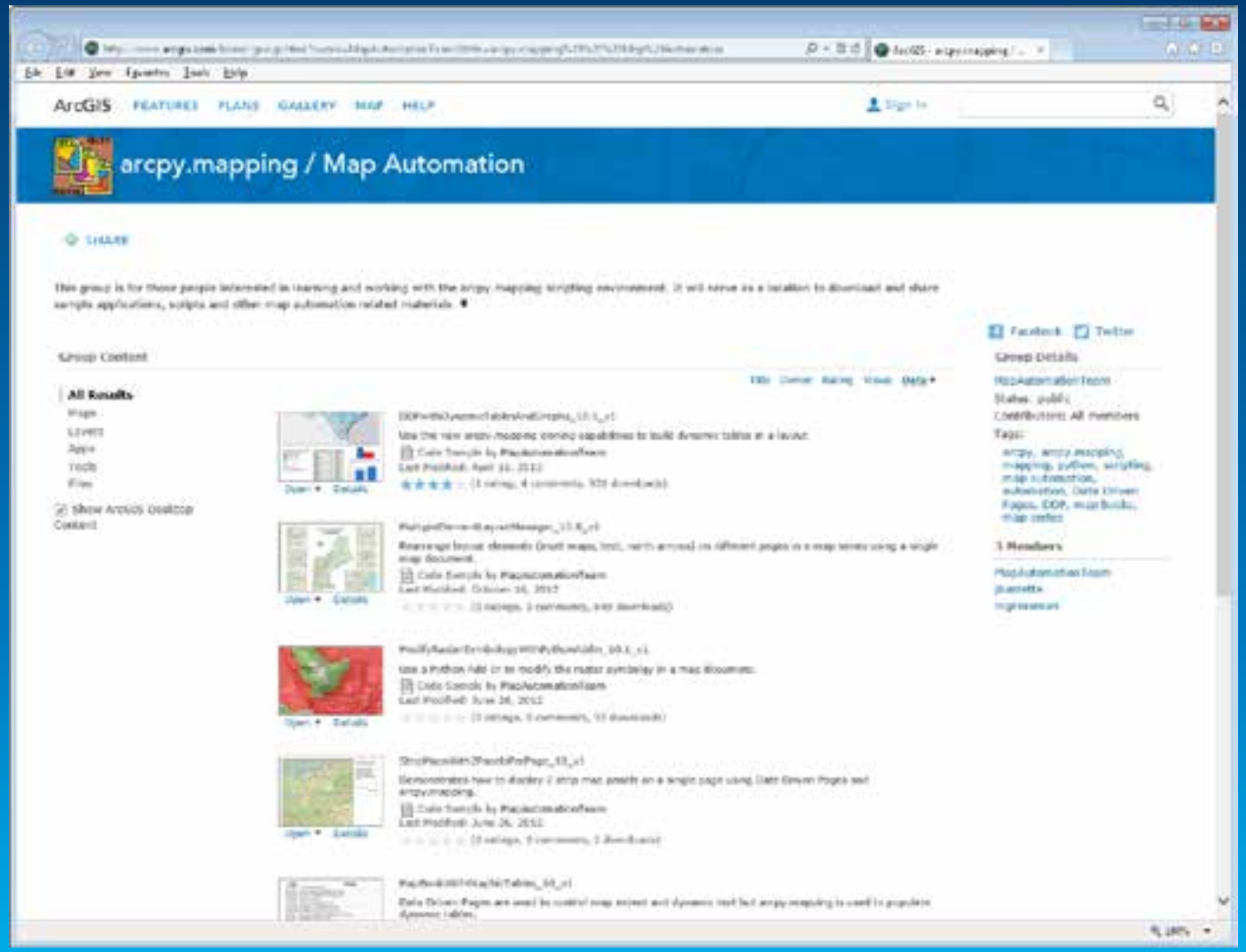
- You can clone text and graphic elements
- This allows you to automate things like dynamic tables

```
vertl = arcpy.mapping.ListLayoutElements(
   mxd, “GRAPHIC_ELEMENT”, “VerticalLine”)[0]
vertl.elementPositionX = xPos;
vertl.elementPositionY = 4
vert1.elementHeight = 3
for line in range(1, numColumns+1):
   vert_clone = vertLine.clone("_clone")
   xPos = xPos + colWidth
   vert_clone.elementPositionX = xPos
```

DEMO

arcpy.mapping group on ArcGIS Online

http://esriurl.com/5915



Performance tips

- Don’t keep calling list functions

```
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

```
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)

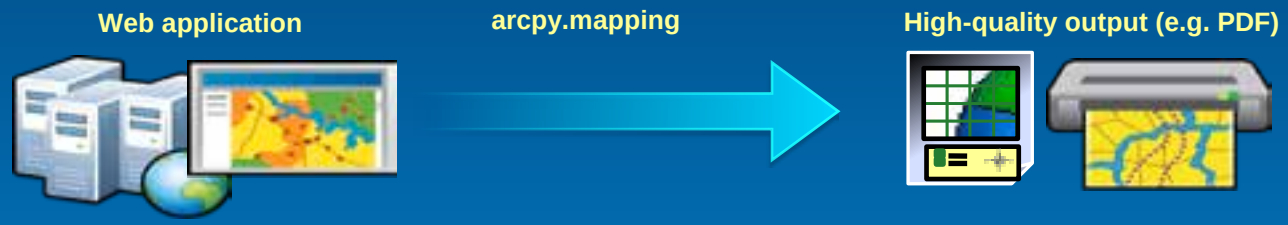
- Use dictionaries

```
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

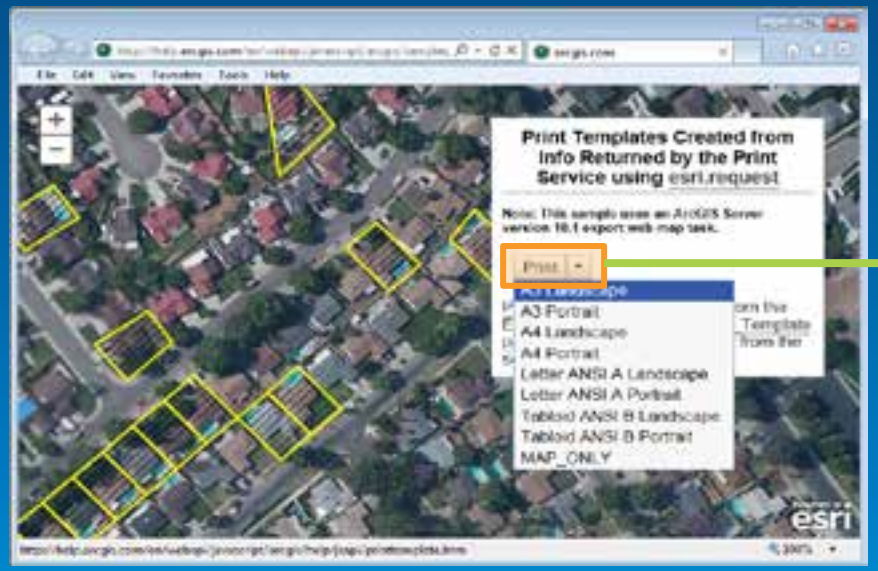
- **ConvertWebMapToMapDocument()**
  - Use with the ArcGIS web APIs for advanced web map printing workflows



- **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**



1

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

2

```
app.printUrl = "http://gilbert:6080/arcgis/rest/services/Utilities/PrintingTools/GPServer/Export%20Web%20Map%20Task";
```

Related Session: Enabling High-Quality Printing in Web Applications (Wednesday @ 12pm – 12:30pm General Theater 3 Exhibit Hall A)

Advanced server printing with arcpy.mapping

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

- **arcpy.mapping method for converting Web Maps to Map Documents:**
  - `ConvertWebMapToMapDocument (webmap_json, {template_mxd}, {notes_gdb}, {extra_conversion_options})`

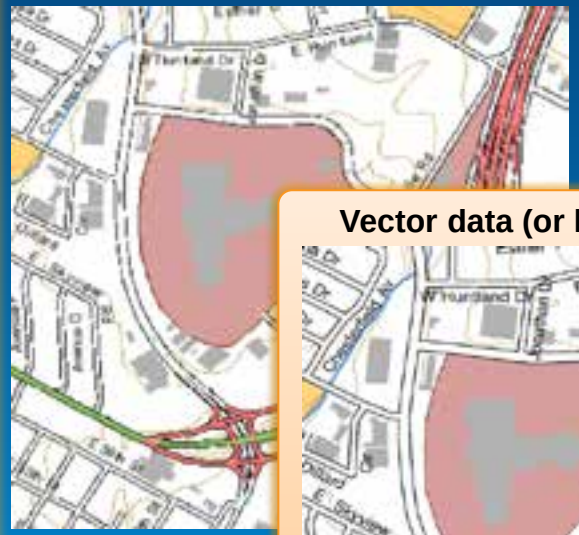
Advanced server printing with arcpy.mapping

- **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

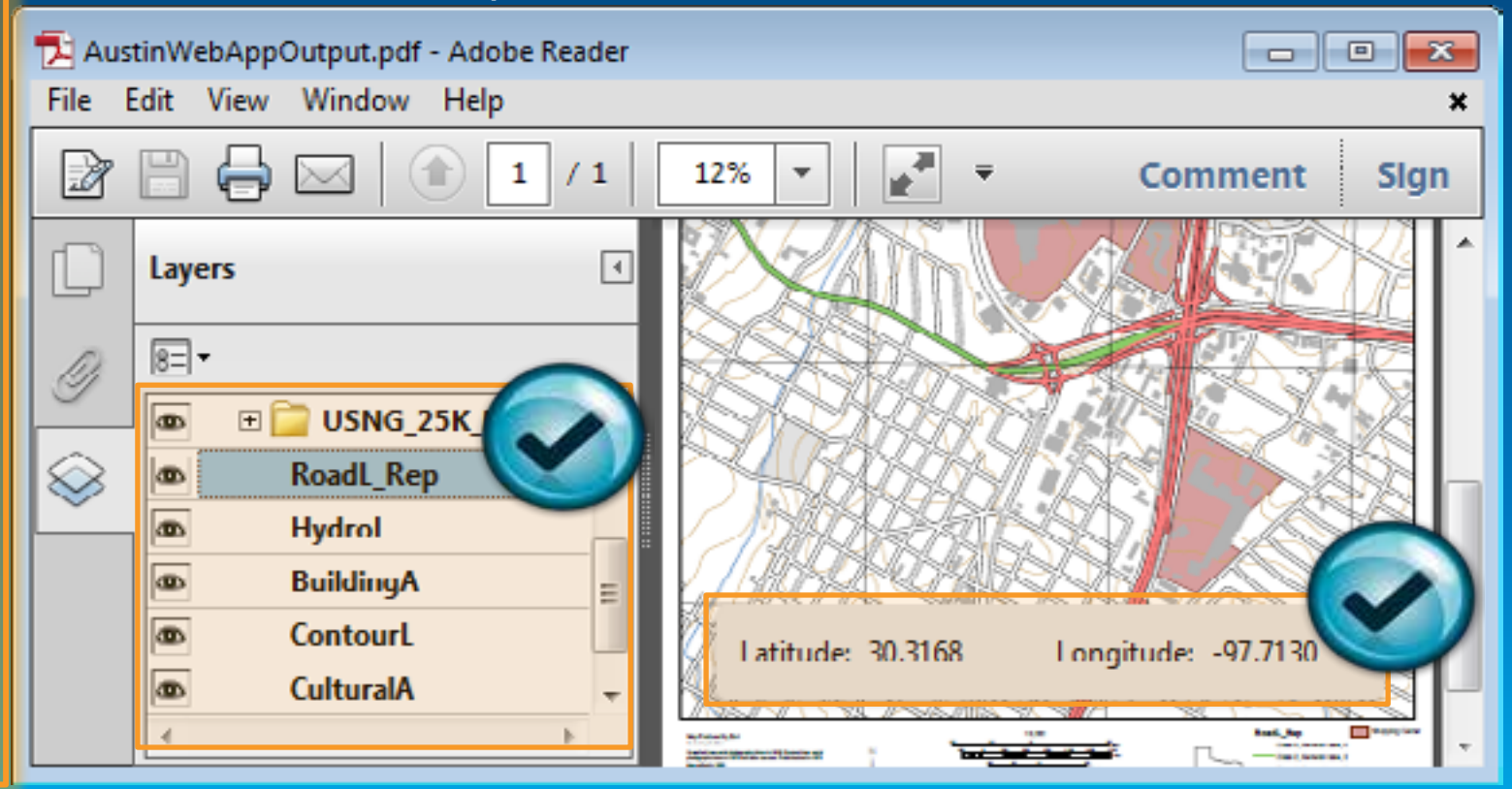
- Output or print vector layers instead of “flat” image of service layers
  - Vector layers will be staged in template map document

Map service tiled cache (low dpi)



Vector data (or high dpi image)

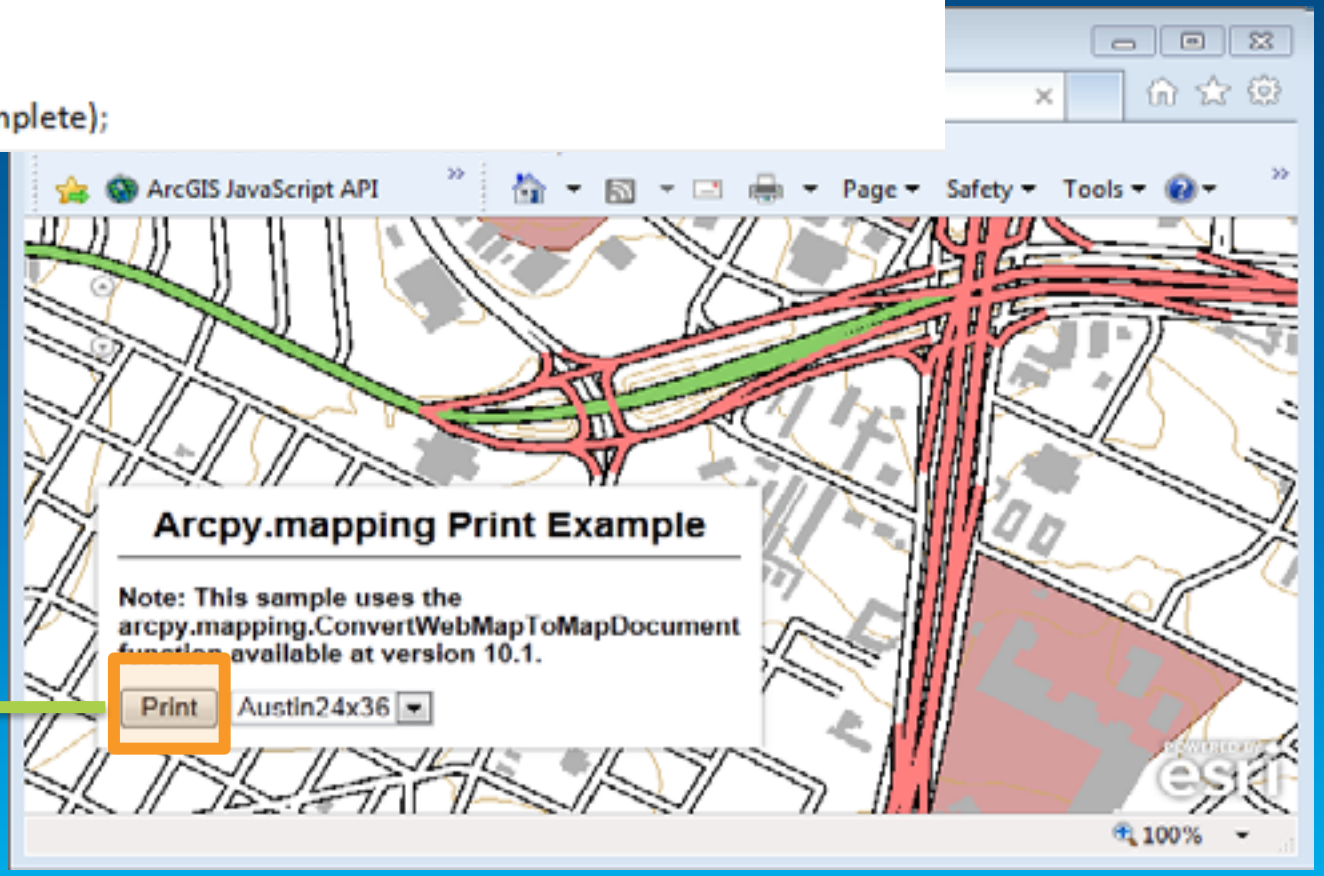
Output PDF viewed in Adobe Reader



Demo: Web app to export vector PDF using arcpy.mapping

- Reference the custom arcpy.mapping based GP service

```
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;
ptemplate.format = "PDF"
params.template = ptemplate;
printTask.execute(params, printComplete);
```



Demo: Web app to export vector PDF using arcpy.mapping

Python code used in custom GP service

```
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;
ptemplate.format = "PDF"
params.template = ptemplate;
printTask.execute(params, printComplete);
```

Get web map JSON

Get template MXD

Create new MXD based on web map

Remove service layers

Export PDF

Output file of job

```
import arcpy, os, uuid

# Input WebMap json
Web_Map_as_JSON = arcpy.GetParameterAsText(0)

# Input Layout template
Layout_Template = arcpy.GetParameterAsText(1)

# The template location in the server registered folder
templatePath = '//gilbert/Austin/Templates'
templateMxd = os.path.join(templatePath, Layout_Template + '.mxd')

# Convert the WebMap to a map document
result = arcpy.mapping.ConvertWebMapToMapDocument(Web_Map_as_JSON, templateMxd)
mxd = result.mapDocument
df = arcpy.mapping.ListDataFrames(mxd, 'Webmap')[0]

# Remove the service layer
# This will just leave the vector layers from the template
for lyr in arcpy.mapping.ListLayers(mxd, data_frame=df):
    if lyr.isServiceLayer:
        arcpy.mapping.RemoveLayer(df, lyr)

# Export the web map to PDF
output = 'WebMap_{}.pdf'.format(str(uuid.uuid1()))
Output_File = os.path.join(arcpy.env.scratchFolder, output)
arcpy.mapping.ExportToPDF(mxd, Output_File, georef_info=True)

# Set the output parameter to be the output file of the server job
arcpy.SetParameterAsText(3, Output_File)
```

Web app to export vector PDF using arcpy.mapping

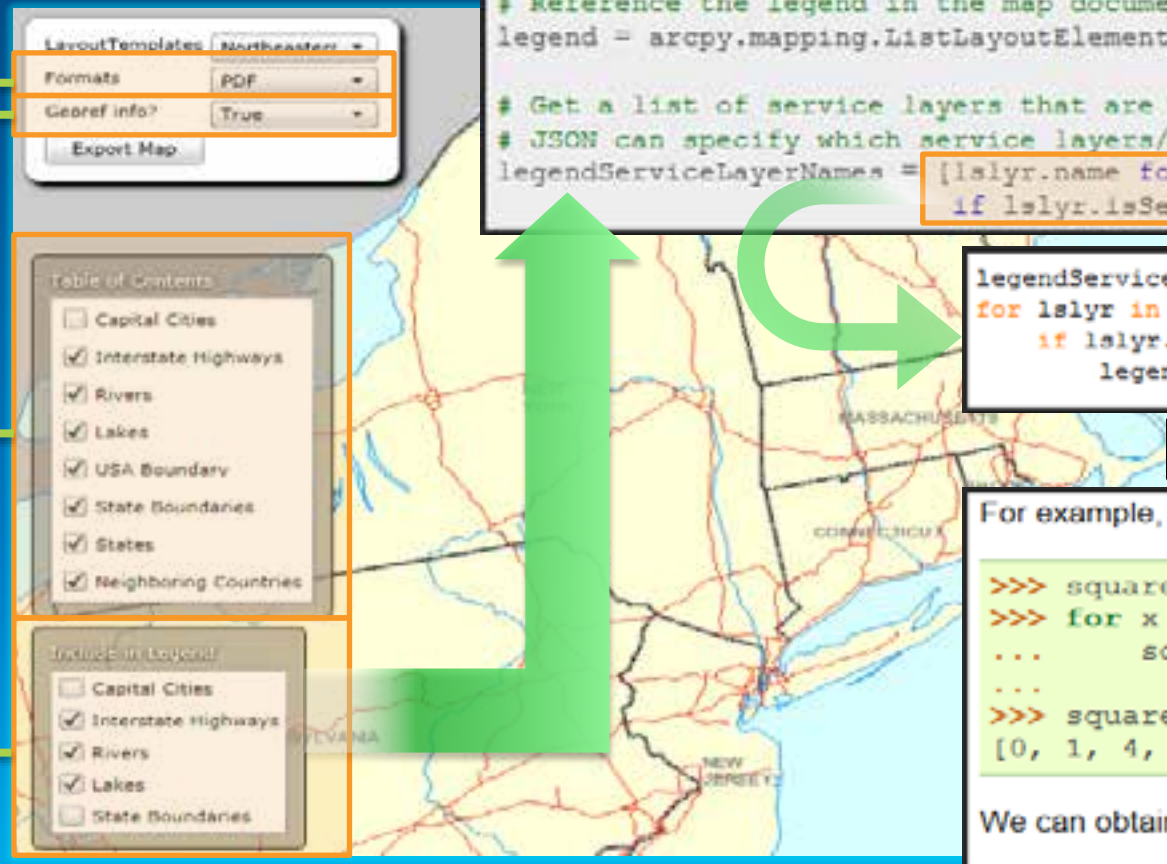
- **Two tutorials in the help:**
  - **Basic vector web map printing: http://esriurl.com/4601**
  - **Advanced web map printing: http://esriurl.com/4602**

Output format

Georeferencing

Layers in map

Layers in legend




```
# 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]
```

```
legendServiceLayerNames = []
for lslyr in legend.listLegendItemLayers():
    if lslyr.isServiceLayer and not lslyr.isGroupLayer:
        legendServiceLayerNames.append(lslyr.name)
```

List Comprehension

For example, assume we want to create a list of squares, like:

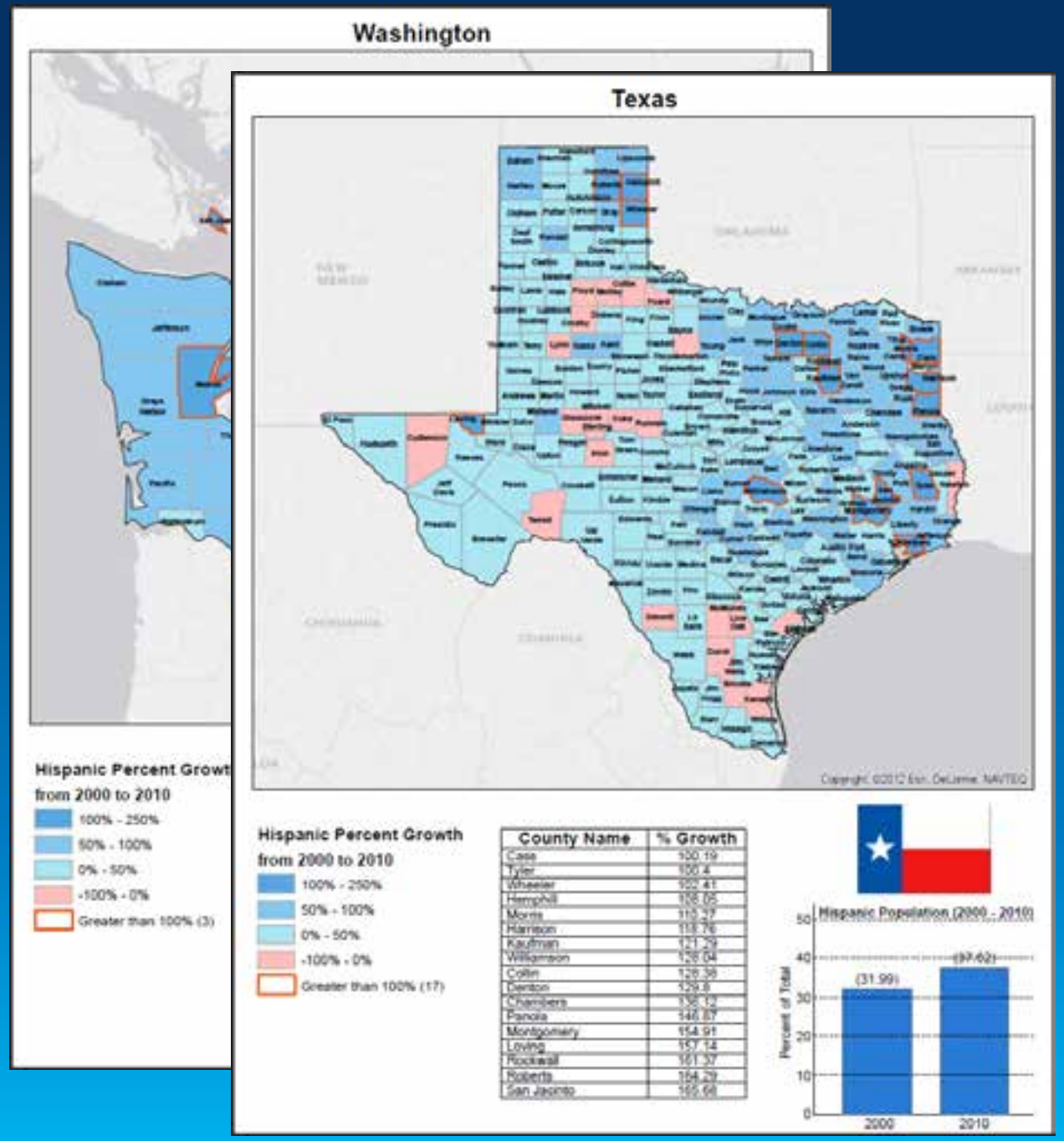
```
>>> squares = []
>>> for x in range(10):
...     squares.append(x**2)
...
>>> squares
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

We can obtain the same result with:

```
squares = [x**2 for x in range(10)]
```

Advanced Server Printing

- **Modify arcpy.mapping scripts authored on Desktop and use them in geoprocesing and print services**



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.

```
# 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)
```

Get JSON Layer Definition from web map

Update vector layer from JSON

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.

Reference & prepare MXDs

arcpy.mapping.MapDocument()

Create SDDraft file
(and optionally edit XML)

arcpy.mapping.CreateMapSDDraft()

Stage and Publish Map Service
(arcpy server GP tools)

arcpy.StageService_server()
arcpy.UploadServiceDefinition_server()

Publishing map services with arcpy.mapping

Sample script: CreateMapSDDraft

Reference MXD

Server connection, service properties, etc.

Create and analyze sddraft for errors, warnings, etc.

Stage and publish Map Service

Don't publish if errors exist

```
import arcpy

# define local variables
wrkspc = 'C:/Project/'
mapDoc = arcpy.mapping.MapDocument(wrkspc + 'counties.mxd')
con = 'GIS Servers/arcgis on MyServer_6080 (publisher).ags'
service = 'Counties'
sddraft = wrkspc + service + '.sddraft'
sd = wrkspc + service + '.sd'
summary = 'Population Density by County'
tags = 'county, counties, population, density, census'

# create service definition draft
arcpy.mapping.CreateMapSDDraft(mapDoc, 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

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
  - DataFrame replaced by Map, MapFrame, and Camera
  - New Layout object
  - Application always refreshes when using CURRENT

Thank you…

Please fill out the session evaluation

First Offering ID: 1307

Second Offering ID: 1405

Online – www.esri.com/ucsessionsurveys

Paper – pick up and put in drop box

Understanding our world.