



**Esri International User Conference | San Diego, CA**  
**Technical Workshops | July 14, 2011**

# **Road Ahead – Python Scripting**

Ghislain Prince

# Python at ArcGIS 10.1

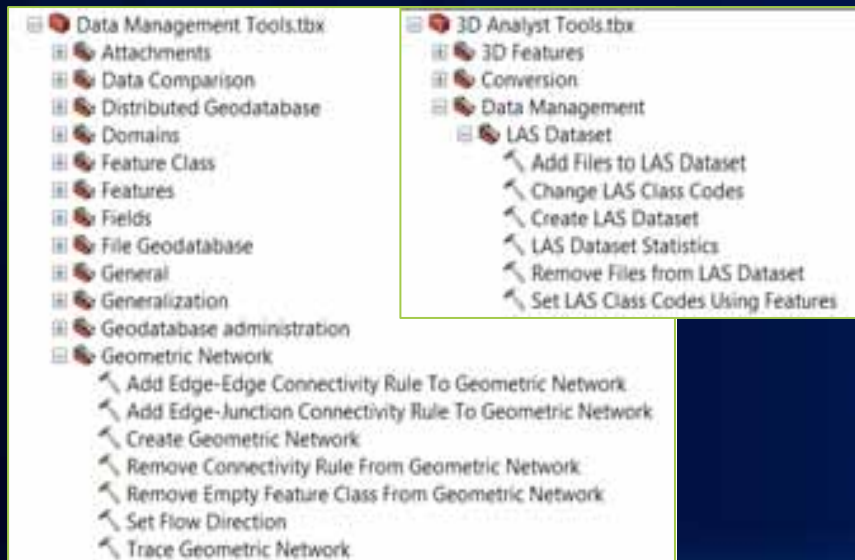
- Upgrade python
- Geoprocessing tools
- Python toolboxes
- Data Access module
  - cursors
  - Edit session
  - NumPy interoperability
- Python add-ins for Desktop
- `arcpy.na`
- `arcpy.mapping`

## Upgrade python

- **ArcGIS 10.1 will ship with Python 2.7.x**
- **Python 3.x later**

# New Tools

- 85 New Tools
- Many existing tools improved



DeKalb County Board

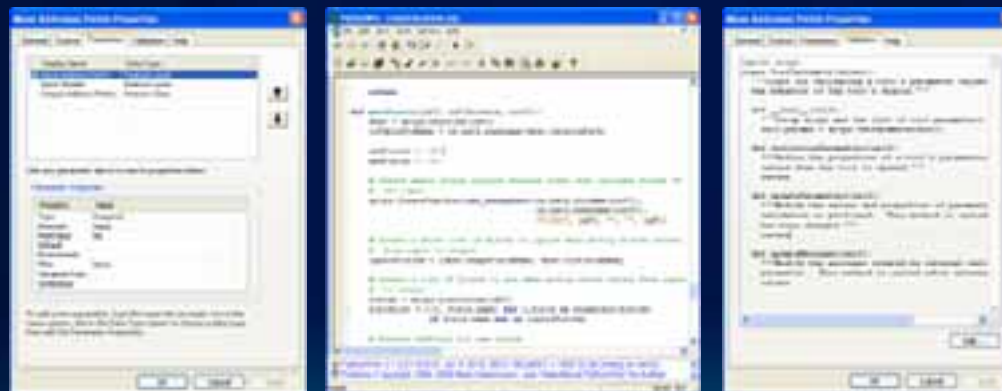
Fulton County Dept. of Health and Wellness/District 3, Unit 2, 2011

# Python toolbox



# Script tool framework

- The script tool framework is geared towards creating tools for novice users
  - Parameters defined through a wizard
  - Validation code that lives in the toolbox
  - Separate source code



# Python toolboxes

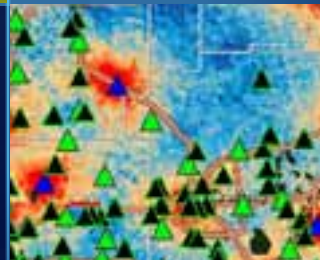
- In a Python toolbox everything is done in Python
  - Easier to create
  - Easier to maintain
- A Python toolbox is an text based file (.pyt) that defines a toolbox and one or more tools
- python toolbox tools are regular gp tools - standard *look and behaviors*



DeKalb County Board

Fulton County Dept. of Health and Wellness/District 3, Unit 2, 04/10/10

# Data access module





## Data access module (arcpy.da)

- *A new Python module for working with data*
- Improved cursors
- Convert data to/from NumPy
- Edit session, edit operation
- Supports versioning and replica workflows

# Cursors

- **Cursors provide record-by-record access**
- **Important part of many workflows**
- **Past performance not fast enough**
  - **Kept algorithms from being developed in Python**
- **SearchCursor – up to 30 times faster**
- **InsertCursor – up to 12 times faster**

# Cursors

- Support **with** statements

```
with arcpy.da.SearchCursor(fc, ['fieldA','fieldB']) as c:  
    for row in c:  
        print(row)
```

- A with statement guarantee close (and release of database locks)

# Edit session

- The Editor class allow use of edit sessions and operations to manage database transactions
- Supports **with** statements
  - Always closes cursor
  - Release database locks and reset iteration

Open an edit session and start an edit operation

```
with arcpy.da.Editor(workspace) as edit:  
    <your edits>
```

Exception—operation is aborted, and edit session is closed without saving

No exceptions—stop the operation and save and close the edit session

## More Editor

Method	Explanation
<b>__enter__ ()</b>	<b>Starts an edit session.</b>
<b>__exit__ ()</b>	<b>If successful, stops editing; saves an edit session. If an exception, stop editing, don't save.</b>
<b>startEditing (workspace)</b>	<b>Starts an edit session.</b>
<b>stopEditing (save_changes)</b>	<b>Stops an edit session.</b>
<b>startOperation ()</b>	<b>Starts an edit operation.</b>
<b>stopOperation ()</b>	<b>Stops an edit operation.</b>
<b>abortOperation ()</b>	<b>Aborts an edit operation.</b>
<b>undoOperation ()</b>	<b>Undo an edit operation (roll back modifications).</b>
<b>redoOperation ()</b>	<b>Redo an edit operation.</b>

# NumPy and ArcGIS

- **NumPy is the defacto Python standard for:**
  - Large array processing
  - Scientific analysis
- **ArcGIS 10 supports converting Rasters to/from NumPy arrays**
- **At 10.1, support for tables and feature classes to/from NumPy arrays**



# Cursors with blobs

- arcpy.da Cursors support blobs

```
import pickle
data = open("c:/images/image1.png", "rb").read()
ic = arcpy.da.InsertCursor("c:/data/fgdb.gdb/fc",
                           ['imageblob'])
ic.insertRow([data])
```

- Blobs return a memoryview

```
import arcpy
sc = arcpy.da.SearchCursor("c:/data/fgdb.gdb/fc",
                           ["imageblob"])
memview = sc.next()[0]
fout = open("c:/images/image1_copy.png", "wb")
fout.write(memview.tobytes())
```

## **Version/Replica support**

- **ListVersions/ListReplicas functions support versioning and replica workflows**
- **Return objects that describe version & replicas**

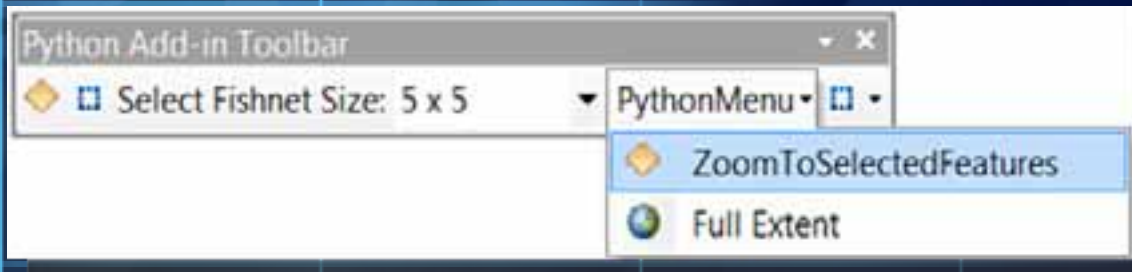
# Version properties (arcpy.Describe)

Property	Explanation	Data Type
access (Read Only)	The version's access permission. <ul style="list-style-type: none"><li>• Private —The version's access permission is private.</li><li>• Public —The version's access permission is public.</li><li>• Protected —The version's access permission is protected.</li></ul>	String
ancestors (Read Only)	The version's ancestors. A list of all the version's that are in the ancestral lineage for the current version. For example, the parent version, the grandparent version, etc. all the way back to the default version.	Version
children (Read Only)	The version's children. A list of all the version's that were created from the current version.	Version
created (Read Only)	The date and time the version was created.	DateTime
description (Read Only)	The version's description.	String
isOwner (Read Only)	True if the current connected user is the owner of this version.	Boolean
lastModified (Read Only)	The last modification of the version.	DateTime
name (Read Only)	The name of the version.	String
parentVersionName (Read Only)	Name of the parent version.	String

# Replica Properties (arcpy.Describe)

Property	Explanation	Data Type
name (Read Only)	The name of the replica.	String
owner (Read Only)	The owner of the data in the replica database.	String
type (Read Only)	The replica access type. <ul style="list-style-type: none"><li>• None —Replica access is undefined. Returned as a None type</li><li>• ReadOnly —Read only child replica (1 way replica).</li><li>• ReadWrite —Read write on both parent and child replica.</li></ul>	String
version (Read Only)	The version the replica was created from.	String
isParent (Read Only)	Indicates the replica role.	Boolean
isSender (Read Only)	Indicates if the replica is a data sender.	Boolean
hasConflicts (Read Only)	Indicates if the replica is in conflict.	Boolean

# Python add-in for Desktop



# Python

- **Python is added to the list of languages for authoring add-ins providing an easy option to extend desktop functionality.**
- **Same file anatomy as existing add-in options (.NET/Java)**
  - **XML file describes the type of customization**
  - **Python script will contain the business logic**
- **NOT the easiest way to put a gp tool on a button**



## Supported Add-in Types

- Buttons & Tools
- Toolbars
- Tool Pallets
- Combo Boxes
- Menus
- Extensions
- **Dockable windows are not supported.**



DeKalb County Board

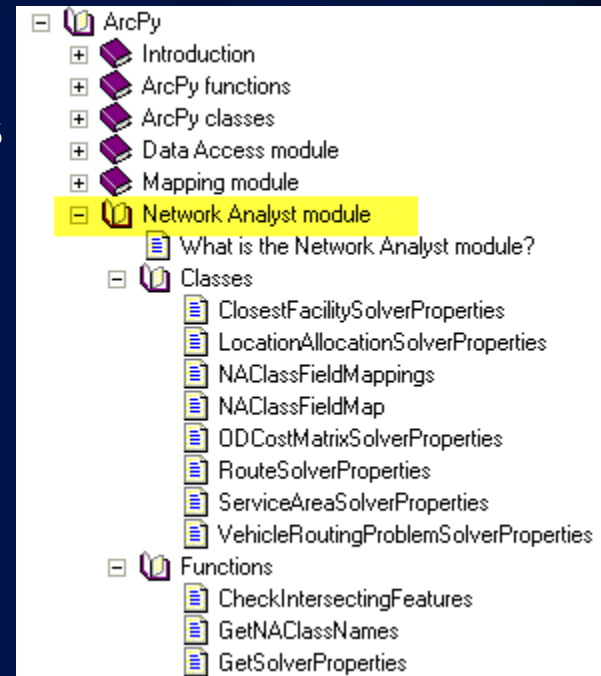
Fulton County Dept. of Health and Wellness/District 3, Unit 2, 100

# Network Analyst



# Network Analyst Module in ArcGIS 10.1

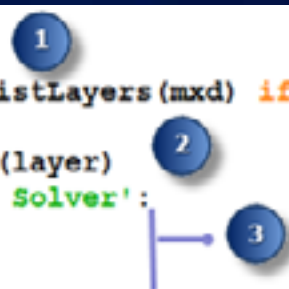
- **Simplify access to Network Analyst functionality from Python**
- **Support editing analysis properties of network analysis layers**
  - **No need to re-create layers**
  - **Speeds up execution**
  - **Simplifies script logic**
  - **Automate workflows from Python window**
- **Provide helper functions and classes to easily use Network Analyst GP tools from Python**



# Editing Network Analysis Layer Properties

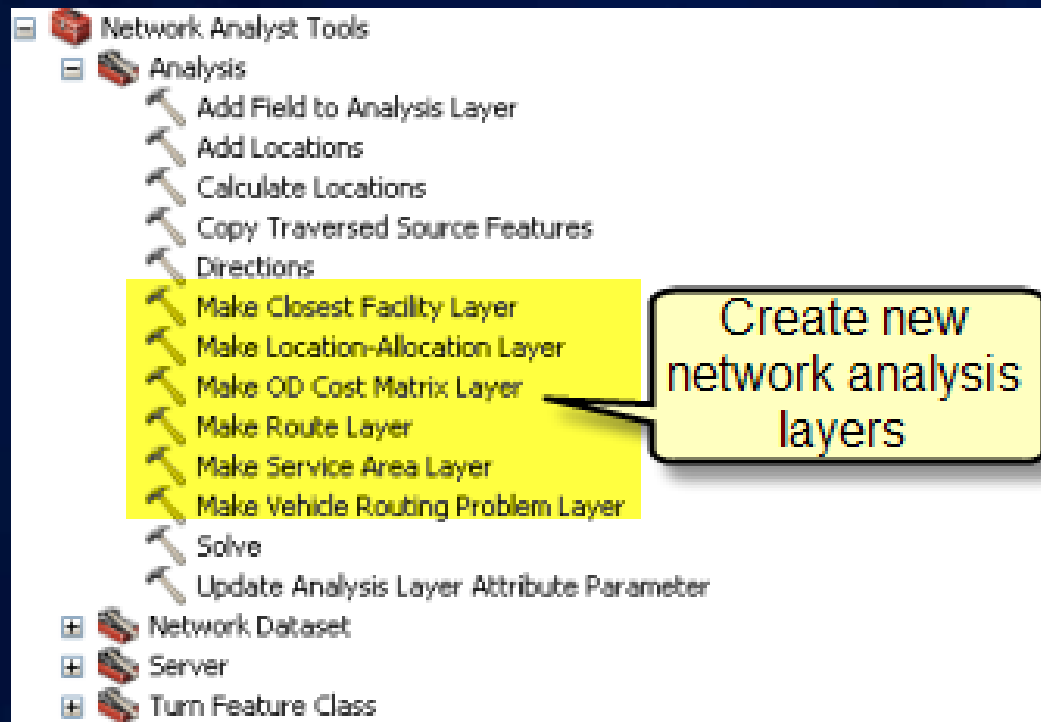
- Edit analysis layers using functions and classes in arcpy.na module
- Common Steps...
  1. Get reference to a network analysis layer object
  2. Get the solver properties object
  3. Update the properties

```
mxd = arcpy.mapping.MapDocument("CURRENT")  
naLayers = [layer for layer in arcpy.mapping.ListLayers(mxd) if layer.isNetworkAnalystLayer]  
for layer in naLayers:  
    solverProps = arcpy.na.GetSolverProperties(layer)  
    if solverProps.solverName == 'Service Area Solver':  
        solverProps.useHierarchy = True  
        solverProps.defaultBreaks = [1,2]
```



# Create Network Analysis Layers

- Create new network analysis layers using GP tools

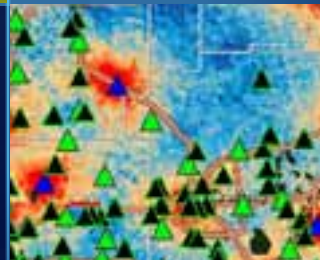




DeKalb County Board

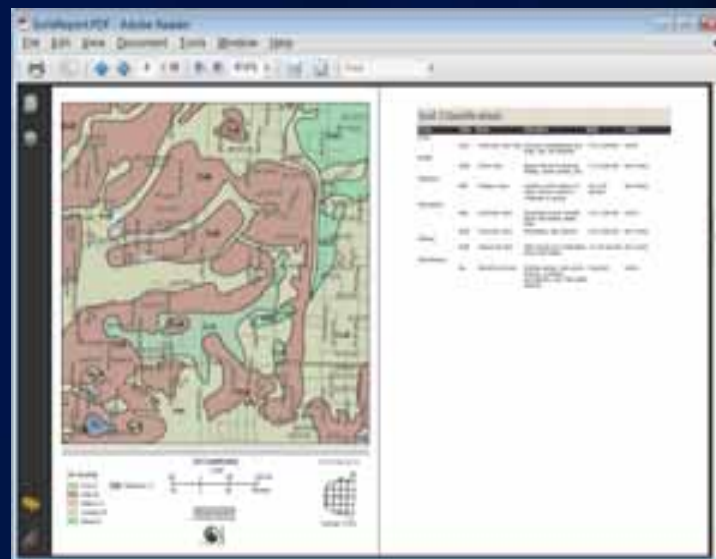
Fulton County Dept. of Health and Wellness/District 3, Unit 2, 06/14/2014

arcpy.mapping



## arcpy.mapping 10.1 – ExportReport

- Export Report
- Built on the ArcMap 10.0 Report Writer engine
- Automate the generation of reports via python
- Integrate report pages into Map Books
- Desktop only functionality

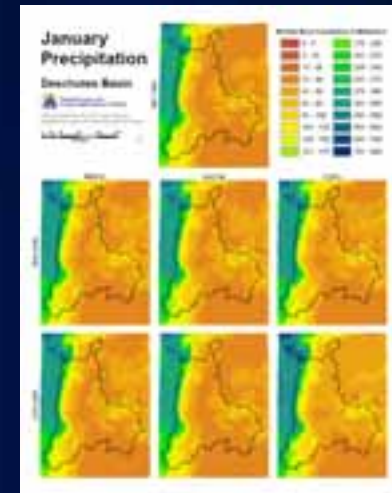


Soils Report

```
arcpy.mapping.ExportReport(lyr, r"C:\Project\Project.rlf",  
    r"C:\Project\Output\ProjectReport.pdf",  
    extent=df.extent)
```

# arcpy.mapping 10.1 – LayerSymbology

- Control renderer classification
- Support for thematic map books
- Access these renderers
  - graduated colors
  - graduated symbols
  - unique values
  - raster classified



```
if lyr.symbologyType == "GRADUATED_COLORS":  
    lyr.symbology.numClasses = 10  
    lyr.symbology.valueField = "POP2007"
```

# arcpy.mapping 10.1 – Other Improvements

- **Layer time**
  - access a layer's time properties
  - enable time on layers
- **Feedback-based improvements to the API**
  - setting text size, setting relative paths, reading page size
- **Bookmarks**
  - interrogate bookmarks and access bookmark extents
  - bookmark-based map book
  - convert bookmarks to feature class

```
for bkmk in arcpy.mapping.ListBookmarks(mxd, data_frame=df):  
    df.extent = bkmk.extent  
    outFile = r"C:\TEMP\BOOKMARKS_OUT\\" + bkmk.name + ".png"  
    arcpy.mapping.ExportToPNG(mxd, outFile, df)
```

- [www.esri.com/sessionevals](http://www.esri.com/sessionevals)

# twitter

---

- <http://twitter.com/arcpy>





esri