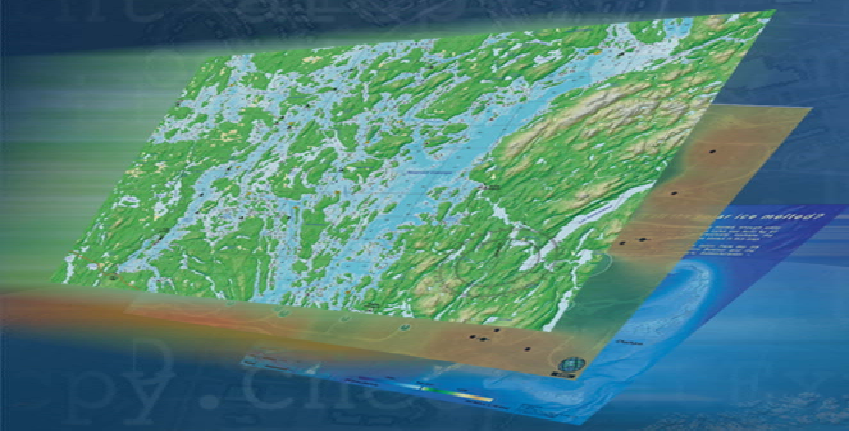
ESRI Developer Summit

March 22–25, 2010
Palm Springs, CA

Getting Started With Python In ArcGIS

Dale Honeycutt
David Wynne



ESRI

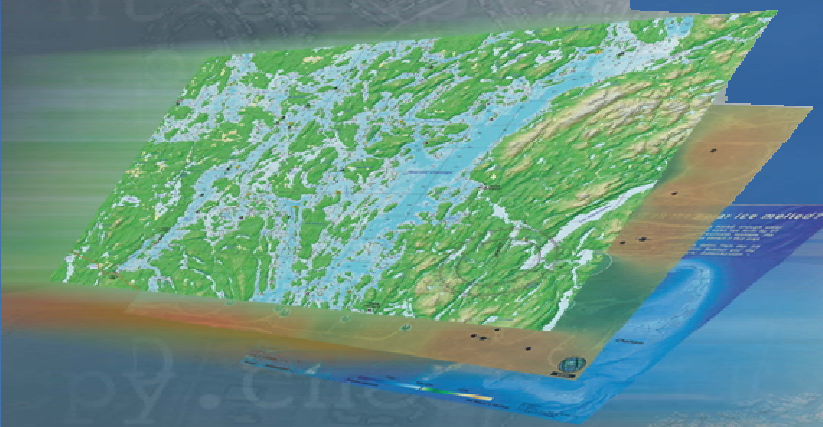
Agenda

- 8:30 – 9:30: Python 101 and ArcPy 101
  - Essentials of the language
  - Using ArcPy to execute geoprocessing tools
  - Python window
- 9:30– 10:00: Script and Script tool basics
  - Creating a standalone script
  - Creating a script tool
- 10:00– 10:15: Map automation
- 10:15 –10:45: Break
- 10:45– 11:00: Raster analysis and Map Algebra
- 11:00– 11:45: Tool Design and Validation

ESRI Developer Summit

March 22–25, 2010
Palm Springs, CA

Python 101



ESRI

Why Python?

- ESRI has embraced Python for ArcGIS 10
- Python is the language that fulfills the needs of our user community
  - Easy to learn
  - Excellent for beginners and experts
  - Suitable for large projects or small scripts
  - Cross-platform

Python 101

- Clear, easy to read syntax
- Easy to use, makes it simple to get started
- Variety of basic data types
  - Including lists and dictionaries
- Comes with a large standard library
- Supports raising and catching exceptions
- Code can be grouped into modules and packages
- Supports object-oriented programming

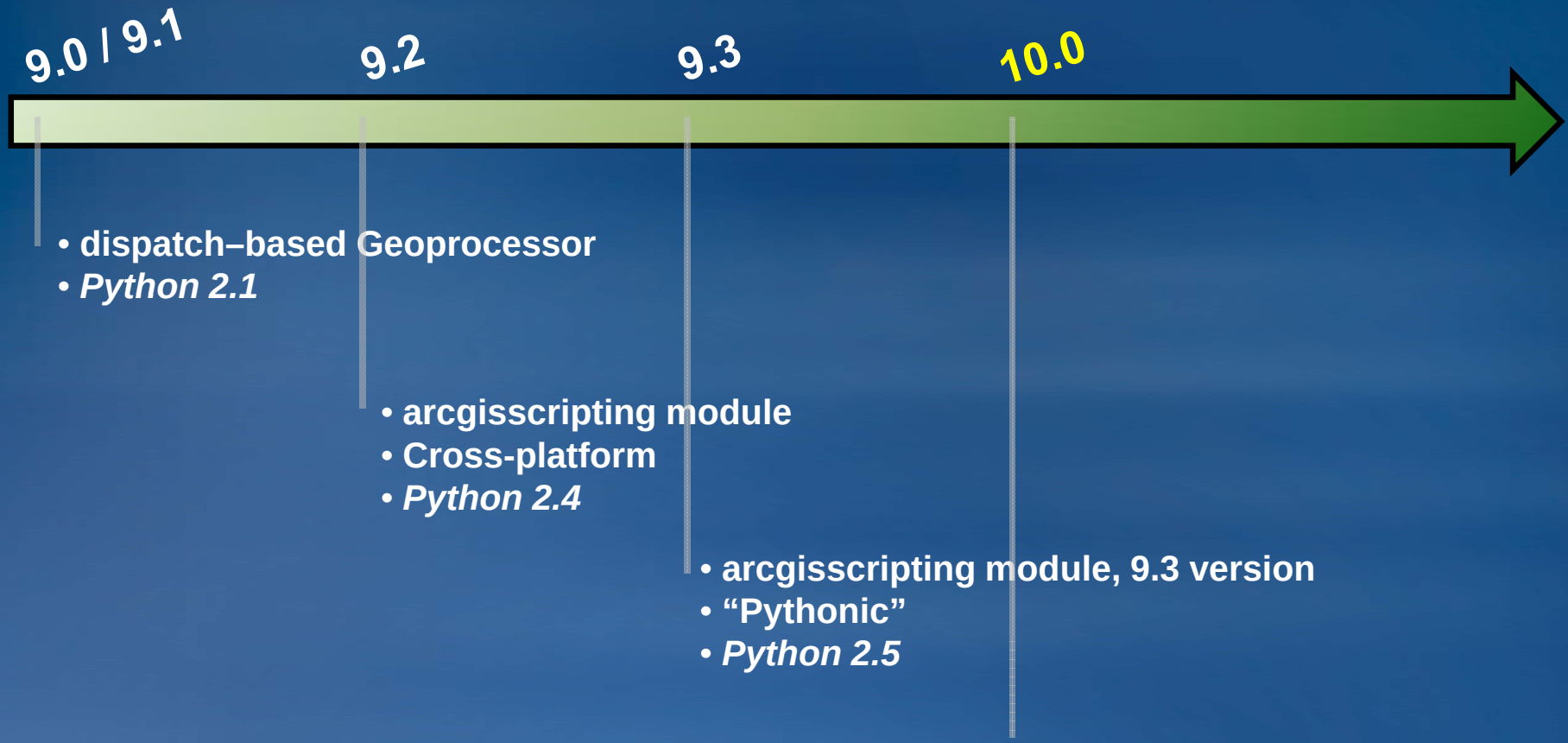
http://www.python.org/about/

Python at 10.0

- **ESRI has fully embraced Python as its language for automation**

1. **ArcPy site-package**
   - **Includes mapping and Map Algebra support**
   - **Successor to the arcgisscripting module**
2. **Python window**
   - **Python access and interaction from within ArcGIS**
3. **Python script tool framework**

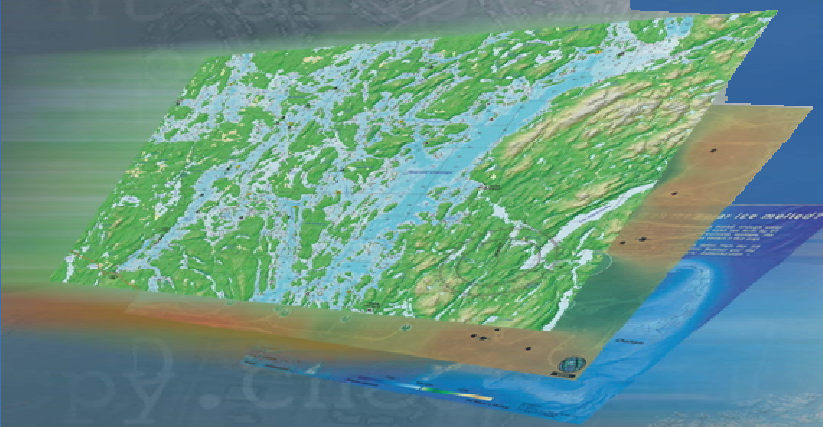
A brief history of Python in ArcGIS



ESRI Developer Summit

March 22–25, 2010
Palm Springs, CA

ArcPy 101



ESRI

What is ArcPy?

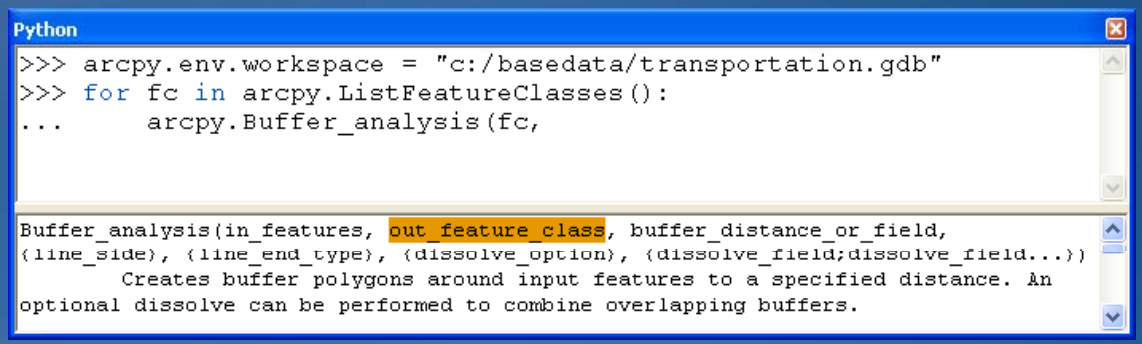
- A cornerstone for automation in ArcGIS
  - data analysis
  - data conversion
  - data management
  - map automation
- ArcPy is a native Python site-package
  - Access to 800+ geoprocessing tools
  - Provides embedded reference documentation for each function, class and module
  - Code completion for ArcGIS components in your favorite Python editor
  - Familiar to arcgisscripting users

ArcPy improvements

- Improved coding experience, such as:
  - Cursors
  - Classes
  - Multi-value parameters can be expressed as Python lists
  - Ability to convert rasters to and from NumPy arrays
- ArcPy is supported by modules, including:
  - A mapping module (arcpy.mapping)
  - A Spatial Analyst module (arcpy.sa) to support map algebra
  - A Geostatistical Analyst module (arcpy.ga)

What is the Python window?

- An embedded Interactive Python window within ArcGIS
  - Can access ArcPy, including tools and environments
  - Can access any other Python functionality,
  - Better code completion and intelligence
- The Python window is for:
  - Testing ideas
  - Experimenting with and learning Python
  - Simple execution of tools
  - Building quick and easy workflows in Python

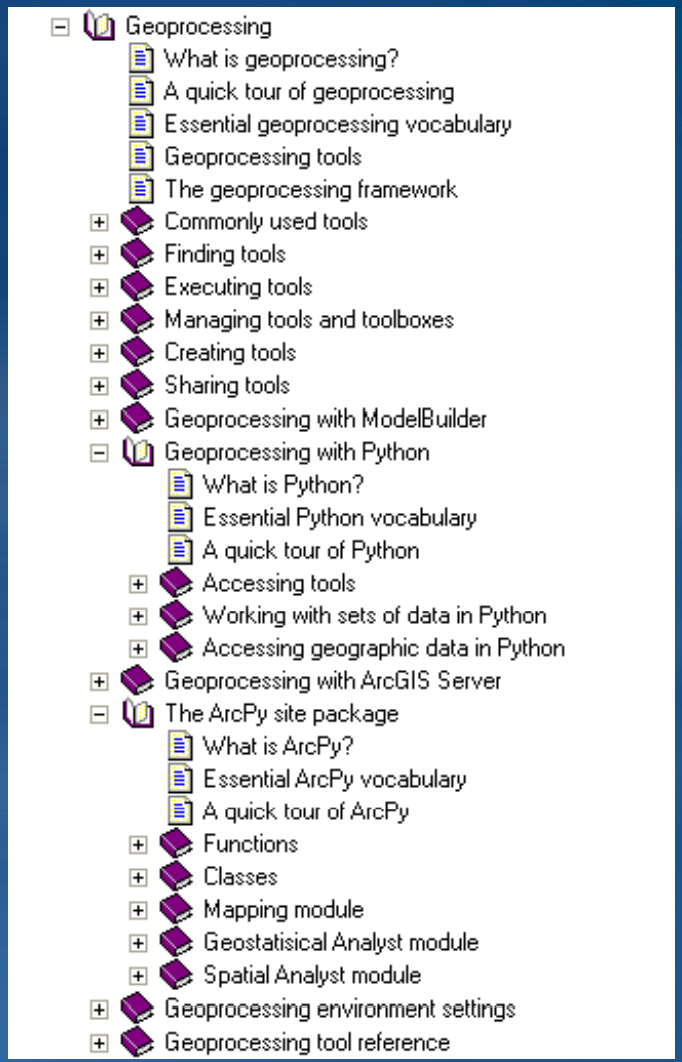


Demo:
A tour of ArcPy
and
the Python window

Getting more help

- **Desktop help**

- **Resource Center**
  - **http://resourcesbeta.esri.com/content/geoprocessing**



Running tools

- Tools are accessed as functions on arcpy
- Environments as properties from arcpy.env class

```
# ~~~ PYTHON CODE ~~~
import arcpy

# Set the workspace
arcpy.env.workspace = "c:/st_Johns/GISData.gdb"

# Execute Geoprocessing tool
arcpy.Intersect_analysis(["roads","urban_area","urban_roads",
                          5, "join")
```

Environments

- Script writers set the environment and tools use it
  - General settings
    - Current Workspace, Output Spatial Reference, Extent
  - Raster analysis settings
    - Cell Size, Mask
  - Many more

arcpy.env.workspace
arcpy.env.outputCoordinateSystem
arcpy.env.extent
arcpy.env.cellSize
arcpy.env.mask

Tool messages

- Executing a tool will produce 3 types of messages.
  - Informative messages (severity = 0)
  - Warning messages (severity = 1)
  - Error messages (severity = 2)

```
# start try block
try:
    arcpy.Buffer("c:/ws/roads.shp", "c:/outws/roads10.shp", 10)

# If an error occurs when running a geoprocessing tool,
#   print the tool messages
except arcpy.ExecuteError:
    print arcpy.GetMessages(2)

# Any other error
except Exception as e:
    print e.message
```

Functions

- Functions perform useful scripting tasks
  - Access geoprocessing tool messages (GetMessages)
  - List data to for batch processing (ListFeatureClasses, ListFields, nine other List functions)
  - Retrieve a dataset's properties (Describe)

```
import arcpy

# Set the workspace for ListFeatureClasses function
arcpy.env.workspace = "c:/test"

# For each fc, create a scratch name and clip
for fc in arcpy.ListFeatureClasses():
    outName = arcpy.CreateScratchName("clipped_" + fc,
                                      "", "featureclass",
                                      arcpy.env.workspace)
    arcpy.Clip_analysis(fc, "boundary", outName)
```

Describe

- Returns an object with dynamic properties
- Allows script to determine properties of data
  - Data type (shapefile, coverage, network dataset, etc)
  - Shape type (point, polygon, line, etc)
  - Spatial reference
  - Extent of features
  - List of fields
- Logic can be added to a script to branch (if statement) based data properties

Describe

```
# Describe a feature class
d = arcpy.Describe("c:/base.gdb/rivers")

# Branch based on the
# input's ShapeType property
if d.shapeType == "Polygon":
 arcpy.FeatureToLine_management(inFC, outFC)
else:
 arcpy.CopyFeatures_management(inFC, outFC)
```

Classes

- Most tool parameters can be easily defined
  - Such as a path or buffer distance
- Some parameters cannot be easily defined with a string
  - Such as a spatial reference or field mapping
- Classes can be used to define parameters

```
prjFile = "c:/North America Equidistant Conic.prj"

# Create a spatial reference using a projection file
spatialRef = arcpy.SpatialReference(prjFile)

# Run CreateFeatureclass using the spatial reference
arcpy.CreateFeatureclass_management(inputWorkspace,
     outputName, "POLYLINE", "", "", "", spatialRef)
```

Cursors

- Cursors allows
  - Iteration over the set of rows in a table
  - Ability to insert new rows into a table
  - Geometry access
- Cursors have three forms
  - Search, Insert and Update

```
import arcpy

# Print road name and road type
for row in arcpy.SearchCursor("D:/data.gdb/roads"):
    print “Road name:” row.roadname
    print “Road type:” row.getValue(“roadtype")

del row
```

Demo:
Converting GPS in XML
to useable Geographic data

A note on Cursor functions

- ArcPy cursors support iteration

At 9.3

```
rows = gp.SearchCursor(myTable)
row = rows.next()
while row:
    print row.GetValue("Rank")
    row = rows.next()
```

At 10

```
for row in arcpy.SearchCursor(myTable)
   print row.GetValue("Rank")
```

Insert Cursor

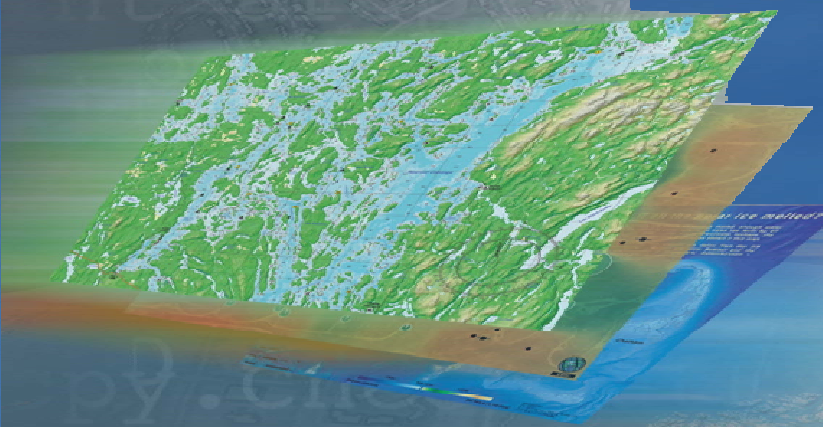
- Cursors support creation of new geometries

```
# Open an insert cursor for the feature class
cur = arcpy.InsertCursor(fc)

# Create array and point objects
ptList = [arcpy.Point(358331, 5273193),
          arcpy.Point(358337, 5272830)]
lineArray = arcpy.Array(ptList)

# Create a new row for the feature class
feat = cur.newRow()

# Set the geometry of the new feature to the
#  array of points
feat.Shape = lineArray

# Insert the feature
cur.insertRow(feat)

# Delete objects
del cur, feat
```

ESRI Developer Summit

March 22–25, 2010
Palm Springs, CA

Creating Python script tools



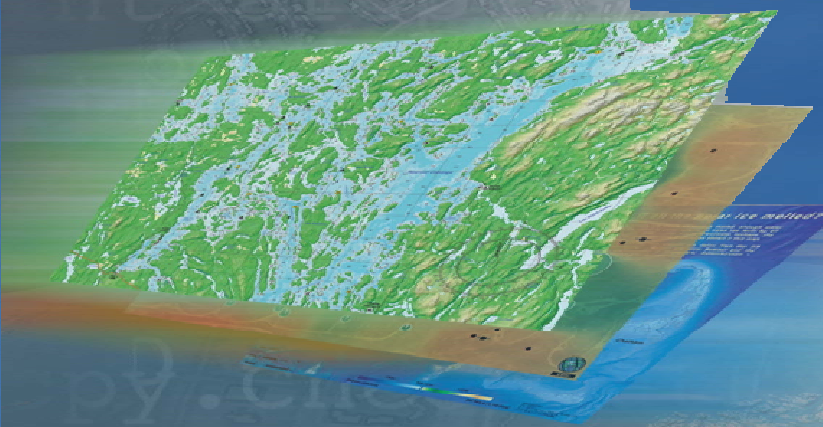
ESRI

Placeholder—building script tool stuff

ESRI Developer Summit

March 22–25, 2010
Palm Springs, CA

arcpy.mapping



ESRI

arcpy.mapping module

- A new mapping module that is part of the Geoprocessing ArcPy site-package
- A python scripting API that allows you to:
  - Manage map documents, layer files, and the data within
    - Find a layer with data source *X* and replace with *Y*
    - Update a layer's symbology across 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/map series

- **Modify map document properties, save changes to a layer file, save changes to the map document**

```
import arcpy
mxd = arcpy.mapping.MapDocument("input.mxd")
df = arcpy.mapping.ListDataFrames(mxd)
df.scale = 24000
df.rotation = 2.7
for lyr in arcpy.mapping.ListLayers(mxd):
    if lyr.name == "Landuse":
        lyr.visible = True
        lyr.showLabels = True
        lyr.saveACopy("output.lyr")
mxd.save()
del mxd
```

Demo:

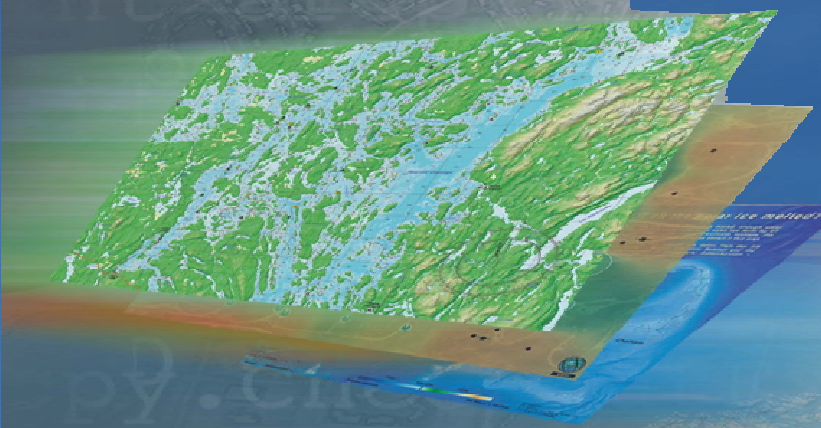
Map automation tools

ESRI Developer Summit

March 22–25, 2010
Palm Springs, CA

BREAK!

See you at 10:45

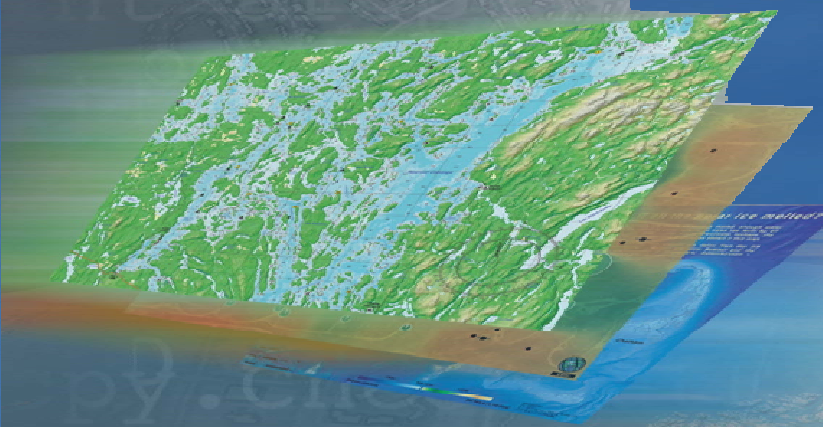


ESRI

ESRI Developer Summit

March 22–25, 2010
Palm Springs, CA

Raster analysis and Map Algebra



ESRI

Raster class

- Returned output from Spatial Analyst tools
  - Used to represent a raster dataset on disk
  - Can be used as inputs to tools and Spatial Analyst Map Algebra expressions
- Supports operators (or arithmetic operations in Map Algebra expressions)
- Has properties and methods for analysis
  - raster.min
  - raster.max
  - raster.save()

Spatial Analyst module

- Spatial Analyst module that integrates Map Algebra into Python
  - Includes all Spatial Analyst tools
  - Supports operators in Map Algebra expressions
  - Helper classes that can be used to support complex parameter
  - Output on the left-side

```
from arcpy.sa import *
demm = Raster("DEM") / 3.28
slpdeg = Slope(demm, "DEGREE")

demfs = FocalStatistics("DEM", NbrRectangle(3,3), "MEAN")
```

Raster Integration

- NumPy is a 3rd party Python library for scientific computing
  - A powerful array object
  - Sophisticated analysis capabilities
- ArcPy Raster objects can be converted to NumPy arrays for analysis
  - RasterToNumPyArray(), NumPyArrayToRaster()

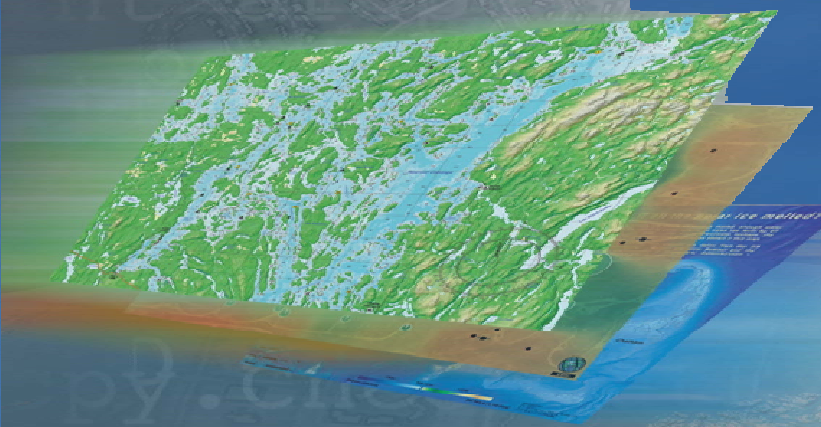
```
inras = "ras100"

# convert raster to Numnpy array.
rasArray = arcpy.RasterToNumPyArray(inras)

# ARRAY SLICING: get the total sum of every third value
#  from every third row of the  raster
sampArray = rasArray[::3,::3]
sum = NUM.sum(sampArray)
print sum
```

Demo:
Working with Map Algebra and NumPy

ESRI Developer Summit

March 22–25, 2010
Palm Springs, CA

Tool Design and Validation



ESRI

Dale’s script tool wizardry – validation