



Esri International Developer Summit
Palm Springs, CA

Working With Raster Data Using Python

Nawajish Noman

Ryan DeBruyn

Outline

- Managing rasters with tools and performing analysis with Map Algebra
- How to access the analysis capability
 - Demonstration
- Complex expressions and optimization
 - Demonstration
- Additional modeling capability using classes
 - Demonstration
- Extending modeling capability using NumPy arrays
 - Demonstration

The problem that is being addressed

- You have a complex modeling problem
- You are working with rasters along with features and tables
- To solve the problem you want to write a script that is
 - Reusable - repeat the workflow with the same or different set of data
 - Dynamic – repeat analysis by using different parameter values
 - Extends capabilities - by taking advantage of 3rd party python packages
 - Performs well – optimized to improve execution speed

The ash borer model

- Movement by flight
 - Fly up to a half mile under its own power
 - Vegetation type and ash density (suitability surface)
- Movement by hitchhiking
 - Roads
 - Camp sites
 - Mills
 - Population
 - Current location of the borer (suitability surface)
- Random movement

Raster analysis

- To prepare and manage raster data
 - Displaying
 - Adding, copying, deleting, etc.
 - Mosaic, Clip, etc.
 - **Raster object**
 - NumPy, ApplyEnvironment, etc.
- To perform the analysis use raster analysis/modeling
 - Spatial Analyst
 - **Map Algebra**

What is Map Algebra

- Simple and **powerful algebra** to execute Spatial Analyst tools, operators, and functions to perform geographic analysis
- The strength is in creating **complex expressions**
- Available through Spatial Analyst module
- Integrated in Python (all modules available)

Importing Spatial Analyst

- Module of ArcPy
- Like all modules must be imported
- To access the operators and tools in an algebraic format the imports are important

```
import arcpy  
from arcpy import env # Analysis environment  
from arcpy.sa import *
```

General syntax

- Map Algebra available through an **algebraic format**
- Simplest form: output raster is specified to the left of an equal sign and the tool and its parameters on the right

```
from arcpy.sa import *  
outRas = Slope(inDem)
```

- Comprised of:
 - Input data
 - Operators
 - Tools
 - Parameters
 - Output

Inputs for analysis

- Rasters
- Features
- Numbers
- Text

- Objects
- Constants
- Variables

```
outRas = Slope(inRaster, "DEGREE", 0.3048)
```

Map Algebra operators

- Symbols for **mathematical operations**
- Many operators in both Python and Spatial Analyst

```
outRas = inRaster1 + inRaster2
```

- Cast the raster (**Raster class constructor**) to indicate operator should be applied to rasters

```
outRas = Raster("inraster1") + Raster("inraster2")  
outRas2 = Raster("inraster") + 8  
outRas3 = inRaster1 + Raster("inraster2") * 8
```

Map Algebra tools

- All the tools that output a raster are available (e.g., Sin, Slope, Reclassify, etc.)

```
outRas = Slope(inRaster, "DEGREE", 0.3048)  
outRas = Aspect("rastername")
```

- Can use any Geoprocessing tools

Tip: Tool names are case sensitive

Tool parameters

- Defines how the tool is to be executed
- Each tool has its own unique set of parameters
- Some are **required**, others are **optional**
- Numbers, strings, and objects (classes)

1 $Slope(in_raster, \{output_measurement\}, \{z_factor\})$

0 `outRas = Slope(inRaster, "DEGREE", 0.3048)`
`outRas = Slope(inRaster, "", 0.3048)`
`outRas = Slope(inRaster)`

2

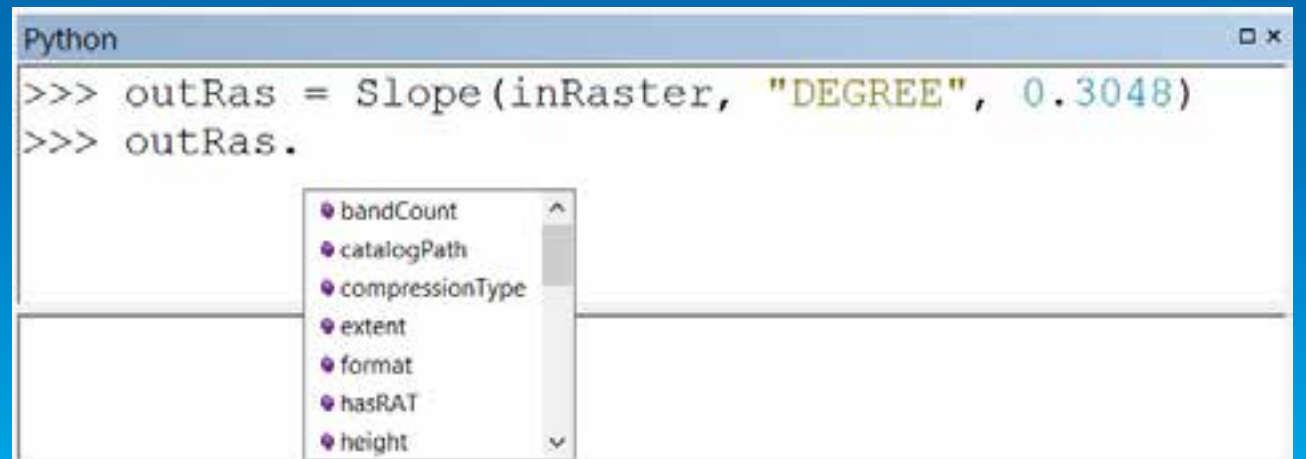
3



Tip: Keywords are in quotes and it is recommended they are capitalized

Map Algebra output

- Returns the results as a **Raster object**
- Object with methods and properties
- The output is **temporary**

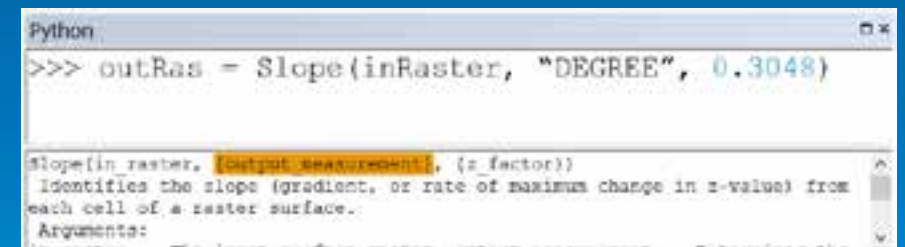


```
Python
>>> outRas = Slope(inRaster, "DEGREE", 0.3048)
>>> outRas.
```

A screenshot of a Python console window. The window title is "Python". The console shows two lines of code: `>>> outRas = Slope(inRaster, "DEGREE", 0.3048)` and `>>> outRas.`. Below the code, a list of properties is displayed, including `bandCount`, `catalogPath`, `compressionType`, `extent`, `format`, `hasRAT`, and `height`. Each property is preceded by a small purple circle icon. The list is enclosed in a box with a scroll bar on the right side.

Access to Map Algebra

- Raster Calculator
 - Spatial Analyst tool
 - Easy to use calculator interface
 - Stand alone or in ModelBuilder
- Python window
 - Single expression or simple exploratory models
- Scripting
 - Complex models
 - Line completion and colors



Demo 1

Data management



- Raster management tools
- Raster Calculator
- Python window
- Simple expressions

Complex expressions

- Multiple operators and tools can be implemented in a single expression
- Output from one expression can be the input to a subsequent expression

Tip: It is a good practice to set the input to a variable and use the variable in the expression

More on the raster object

- A **variable** with a pointer to a dataset
- Output from a Map Algebra expression or from an existing dataset
- The associated dataset is **temporary** (when created from Map Algebra) but has a save method
- A series of properties describing the associated dataset
 - Description of raster (e.g., number of rows)
 - Description of the values (e.g., mean)

Optimization

- A series of local tools (Abs, Sin, Cell Statistics, etc.) and operators can be optimized
- Work on a per-cell basis
- When entered into a single expression each tool and operator is processed on a per cell basis

The iterative aspects of the ash borer model

- Movement by hitchhiking
 - Based on highly susceptible areas
 - Nonlinear decay
 - Random points and check susceptibility
- Movement by flight
 - Depends on the year how far it can move in a time step
 - “Is there a borer in my neighborhood”
 - “Will I accept it” – suitability surface
- Random movement
 - Nonlinear decay from known locations (NumPy array)

Demo 2

Movement by hitchhiking



- Roads, Campsites, Mills, Population, and current location (suitability)
- Complex expressions
- Raster object
- Optimization

Classes

- Objects that are used as parameters to tools
 - Varying number of arguments depending on the selected parameter choice (neighborhood type)
 - The number of entries into the parameters can vary depending on the specific situation (a remap table)
- More flexible
- Query the individual arguments

Classes (contd.)

- Creating

```
nbr01 = NbrCircle(4, "MAP")
```

- Querying

```
radius01 = nbr01.radius
```

- Changing arguments

```
nbr01.radius = 6
```

Vector integration

- Feature data is required for some Spatial Analyst Map Algebra
 - IDW, Kriging, etc.
- Geoprocessing tools that operate on feature data can be used in an expression
 - Buffer, Select, etc.

The iterative aspects of the ash borer model

- Movement by hitchhiking
 - Based on highly susceptible areas
 - Nonlinear decay
 - Random points and check susceptibility
- Movement by flight
 - Depends on the year how far it can move in a time step
 - “Is there a borer in my neighborhood”
 - “Will I accept it” – suitability surface
- Random movement
 - Nonlinear decay from known locations (NumPy array)

Demo 3

Movement by flight



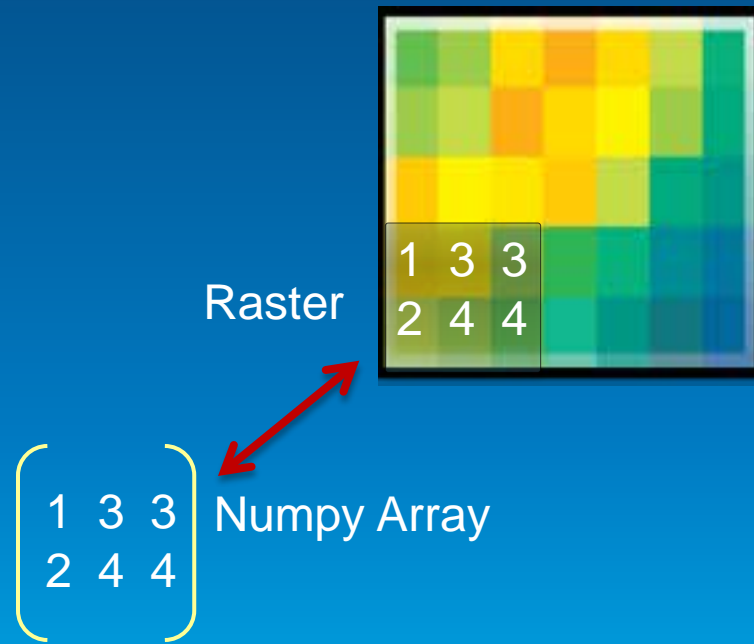
- 20 km per year
- Vegetation type/ash density (suitability)
- Classes
- Using variables
- Vector integration

NumPy

- An extension package to Python
 - Adds support for large, multi-dimensional arrays and matrices
 - Provides a large library of high-level mathematical functions
- Can be used to extend raster analysis capabilities by creating
 - custom functions
 - custom tools
- Access the wealth of free tools built by the scientific community
 - Clustering
 - Filtering
 - Linear algebra
 - Optimization
 - Fourier transformation
 - Morphology

NumPy Arrays

- Two functions
 - RasterToNumPyArray
 - NumPyArrayToRaster



The iterative aspects of the ash borer model

- Movement by flight
 - Depends on the year how far it can move in a time step
 - “Is there a borer in my neighborhood”
 - “Will I accept it” – suitability surface
- Movement by hitchhiking
 - Based on highly susceptible areas
 - Nonlinear decay
 - Random points and check susceptibility
- Random movement
 - Nonlinear decay from known locations (NumPy array)

Demo 4

The random movement



- Random movement based on nonlinear
- decay from existing locations
- Custom function
- NumPy array

Summary

- When the problem become more complex you may need additional capability provided by Map Algebra
- **Map Algebra** powerful, flexible, easy to use, and integrated into Python
- Accessed through: Raster Calculator, Python window, ModelBuilder (through Raster Calculator), and scripting
- Raster object and classes
- Create models that can better **capture interaction** of phenomena
- Demos are available online at <http://www.arcgis.com/home/item.html?id=5a42c7c368e444818615c584fdddb13a>



Understanding our world.