

Python Raster Function: Custom On-the-fly Analysis

Feroz Abdul Kadar

Jamie Drisdelle

Raster Functions



What's a Raster Function?

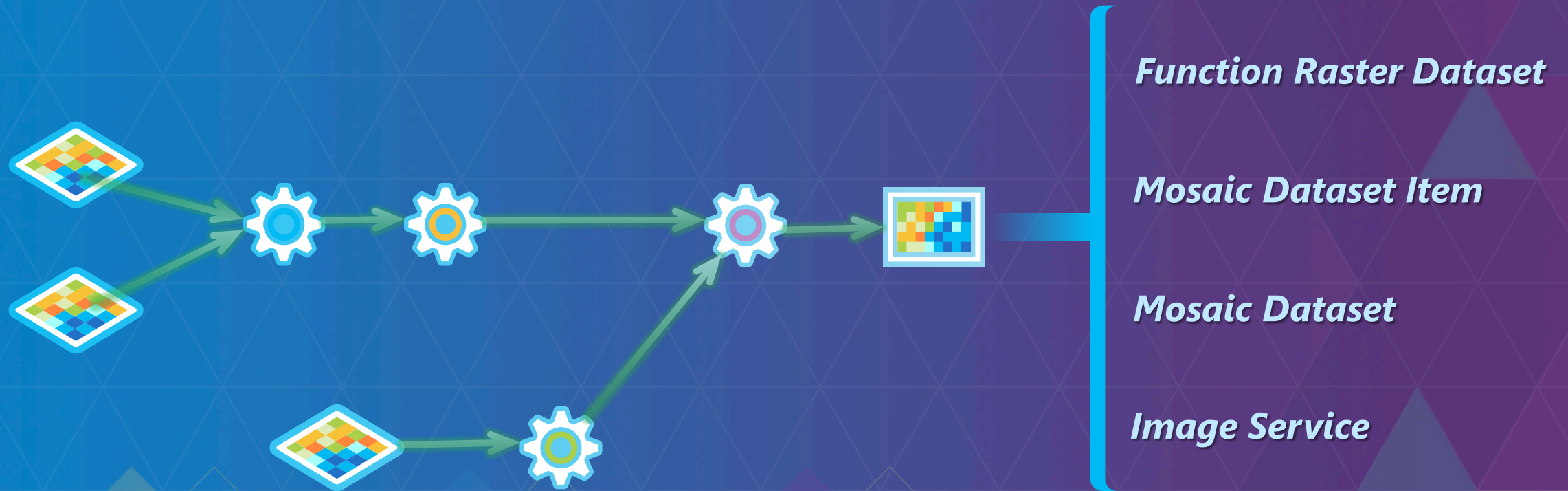
The Fundamentals

- Mapping of one raster to another.
- Loosely $I : \mathbb{N}^3 \rightarrow \mathbb{R}$ $F : (I, \Phi) \rightarrow \mathbb{R}$
- Transient: on the fly.
- Different from a geoprocessing tool.

*... a **transformation** of one raster into another.*

What's a Raster Function *Chain*?

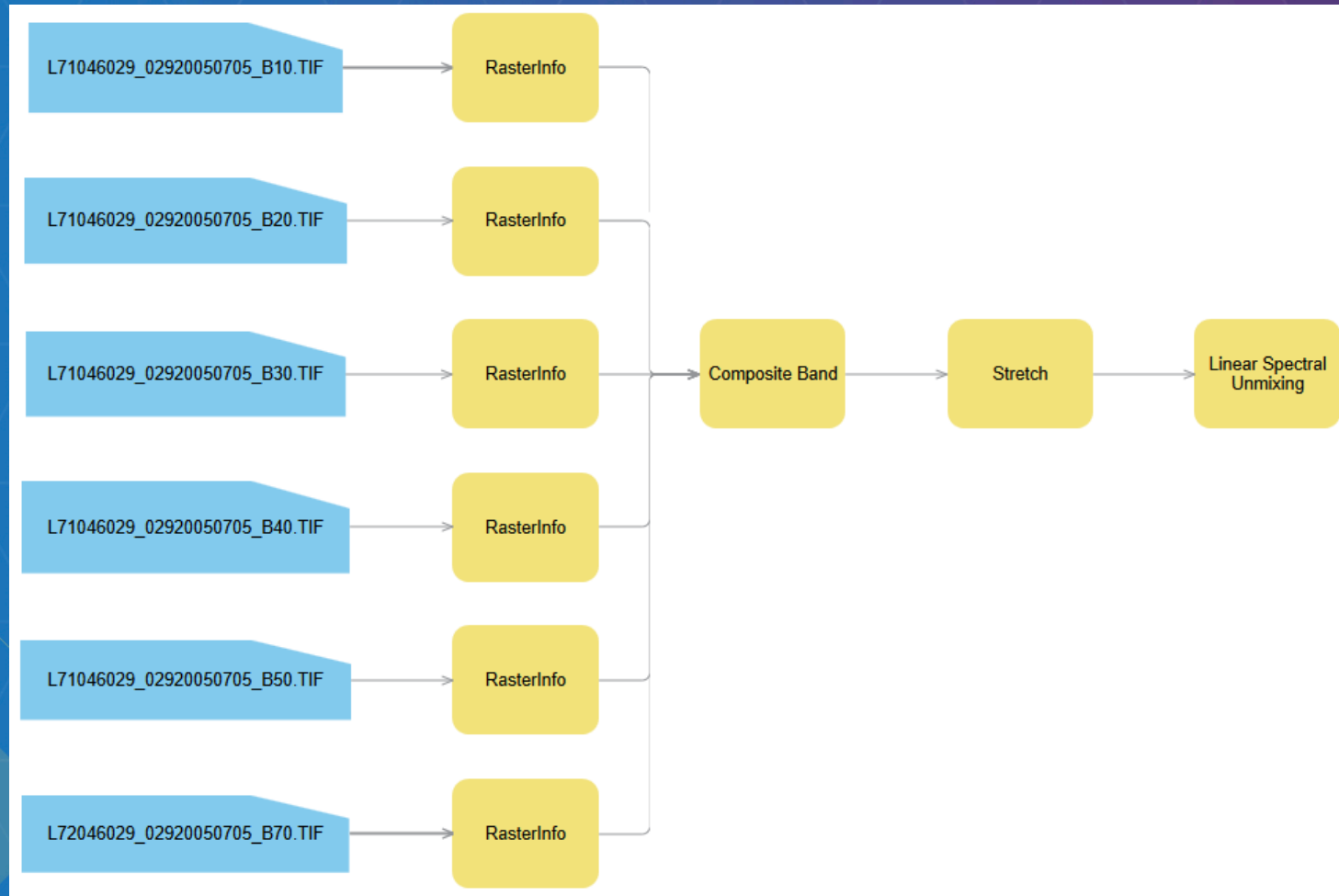
The Basic Model



*A raster function “chain” encapsulate your algorithm as a **tree** of functions.*

What's a Raster Function *Chain*?

The Basic Model



Python Raster Functions



What's a *Python* Raster Function?

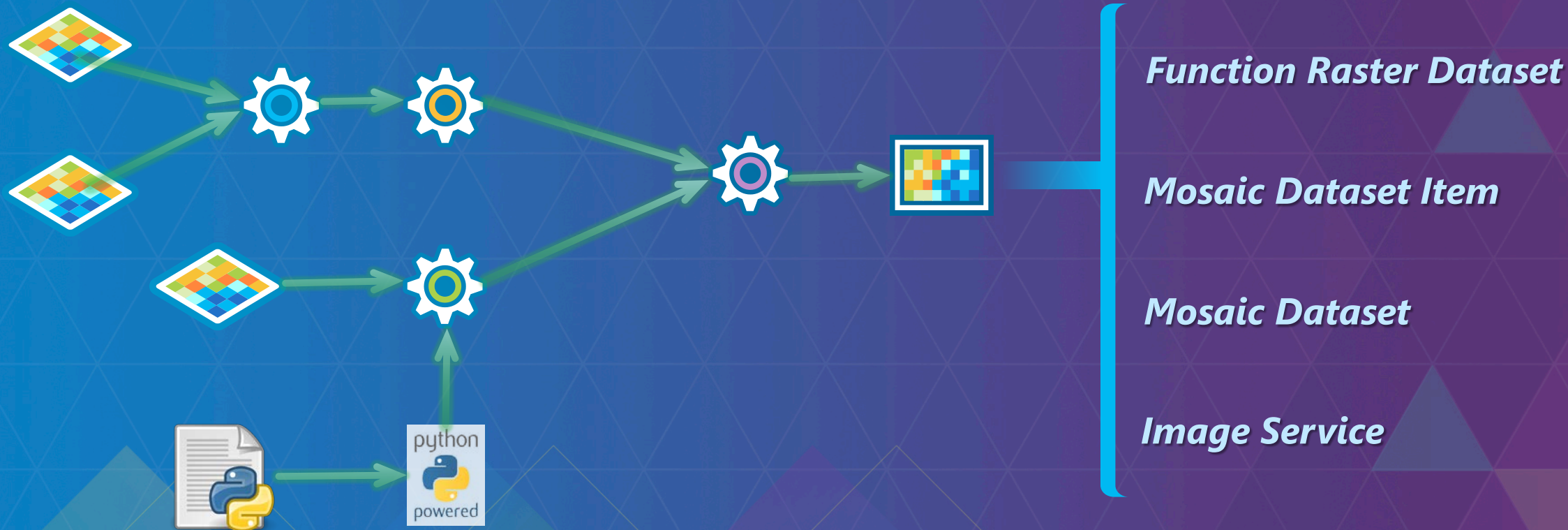
The Extended Model

- Natural evolution of the raster function's COM API—usually implemented in .NET
- Implement a raster function from the comfort of your *Python module*.
- *Architecture*: Module loaded by an adapter—Python-aware *and* a first-class participant in the function chain.

Your Python module—assisted by ArcGIS—is a raster function.

What's a *Python* Raster Function?

The Extended Model



Motivation

Why Raster Functions in Python?

- Extend ArcGIS—participate in a *raster function chain*.
- Primary pipeline for image data in ArcGIS—*processing, analyzing, and visualizing... on the fly*.
- Portable. Reusable. Dynamic. Fast. Scalable.
- Why Python?
 - Friendly & easy to learn. “readability first”. “batteries included”.
 - Huge collection of libraries. Vibrant community of *Pythonistas* and *Pythoneers*.
 - “...de facto ***superglue*** language for ***modern scientific computing***.”
 - “...tools for almost every aspect of scientific computing are *readily available in Python*.”

An Example

*Raster Function in Python: **Linear Spectral Unmixing***

Courtesy: Jacob Wasilkowski. [@jwasil](https://twitter.com/jwasil)



Linear Spectral Unmixing

Introducing a Sample Use Case

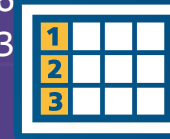
- Determines *abundance* of certain material in a particular pixel.
- Typically performed on lower resolution imagery (like Landsat)
- Input training *endmember* spectra required for each class

Linear Spectral Unmixing

The Process



Shadow: [70.05629, 27.24081, 25.31275, 24.17432, 31.77904, 17.82422]
Veg: [65.46086, 30.09995, 26.27376, 117.45741, 76.96012, 26.25062]
NPV: [74.74029, 32.06931, 35.57350, 32.66022, 72.63062, 60.51104]
Soil: [143.65580, 79.30271, 102.82176, 93.176.57705, 117.49280]



Soil
Vegetation
Shadows
Non-photosynthetic vegetation

Demo

*A Python Raster Function in Action: **Linear Spectral Unmixing***

***Courtesy:** Jacob Wasilkowski. [@jwasil](https://twitter.com/jwasil)*



The API

How do I create Raster Function in Python?

- How does ArcGIS Desktop or Server *interact* with my raster function?
 - Get started—step-by-step guide
 - Real-world or reference implementations—excellent springboard
 - Well-documented API reference
- What additional libraries are needed? How complicated is it?
 - Lightweight design—no external dependencies outside of NumPy to begin with.
 - ArcGIS' *adapter* provides assistance—opt out to take control of specific aspects.

*Create a new raster function using **simple** and **familiar Pythonesque** constructs.*

Hello, World!

```
import numpy as np

class HelloWorld():
    def __init__(self):
        self.name = "Hello World Function"

    def getParameterInfo(self):
        return [{
            'name': 'r',
            'dataType': 'raster'
        }]

    def updatePixels(self, tlc, shape, props, **pixelBlocks):
        r = pixelBlocks['r_pixels'] + 10
        pixelBlocks['output_pixels'] = r.astype(props['pixelType'])
        return pixelBlocks
```

The API

`__init__`

- customize our function object—a specific instance of our class—as soon as it's created.
- Define raster function *name* & *description*.

Linear Spectral Unmixing Properti... ×

General | Parameters | Variables

Name
Linear Spectral Unmixing

Description
Performs linear spectral unmixing for a multiband raster.

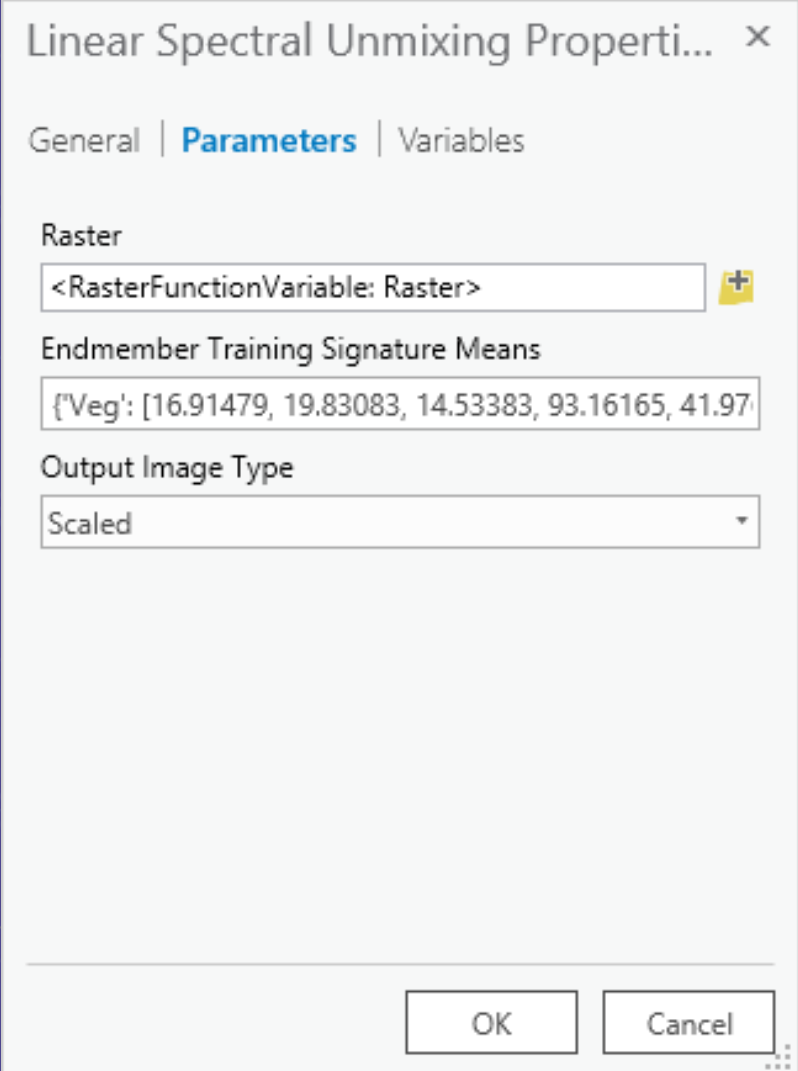
Output Pixel Type
32 Bit Unsigned ▾

OK Cancel

The API

getParameterInfo()

- Define all *input parameters* to the function.
- For each parameter, define:
 - Name (Identifier)
 - Display Name
 - Long Description
 - Data Type
 - Default Value
 - Required vs Optional
 - ...



Linear Spectral Unmixing Properti... x

General | **Parameters** | Variables

Raster
<RasterFunctionVariable: Raster> +

Endmember Training Signature Means
{'Veg': [16.91479, 19.83083, 14.53383, 93.16165, 41.97]}

Output Image Type
Scaled ▾

OK Cancel

The API

getConfiguration()

- How are input rasters read—Padding, Mask, ...?
- How's the output raster constructed—inherit NoData, Metadata, ...?
- **Given:** Nothing.
- **Returns:** dictionary containing configuration attribute values.

The API

selectRasters()

- Define a subset of input rasters.
- Pixels read from *selected* rasters.
- **Given:** properties of the requested pixel block, all scalar parameter values.
- **Returns:** names of selected rasters.



The API

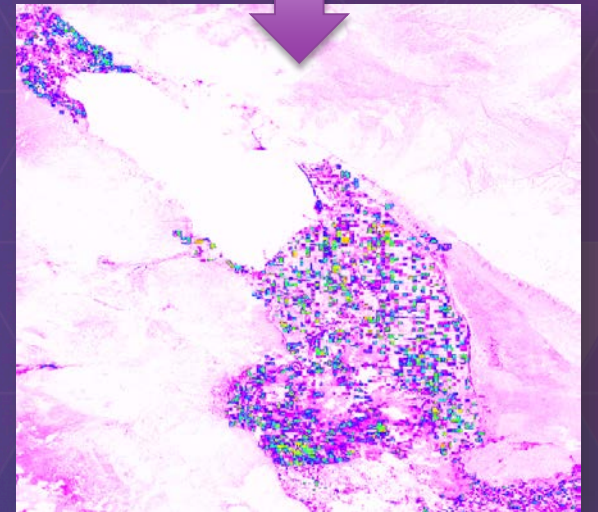
updateRasterInfo()

- Defines the output raster.
- Invoked each time a dataset containing the Python raster function is initialized.
- **Given:** Raster info associated with all input rasters.
- **Returns:** *Raster Info* of the output raster.

The API

`updatePixels()`

- Workhorse of the raster function. Process Pixels.
- **Given:**
 - Expected pixel-block size+location
 - output raster properties (map space)
 - pixels+mask of selected input rasters
- **Returns:** Pixels+mask of requested pixel block.



The API

updateKeyMetadata()

- Create or update dataset- or band-level metadata.
- **Given:**
 - property names
 - band index
 - current key metadata values
- **Returns:** updated values of given properties

The API

isLicensed()

- ***Given:***
 - info on parent product,
 - context of execution.
- ***Returns:***
 - OK to Run (Boolean)—Licensed to execute or not?
 - Expected product level and extension.

Demo

Code Walkthrough

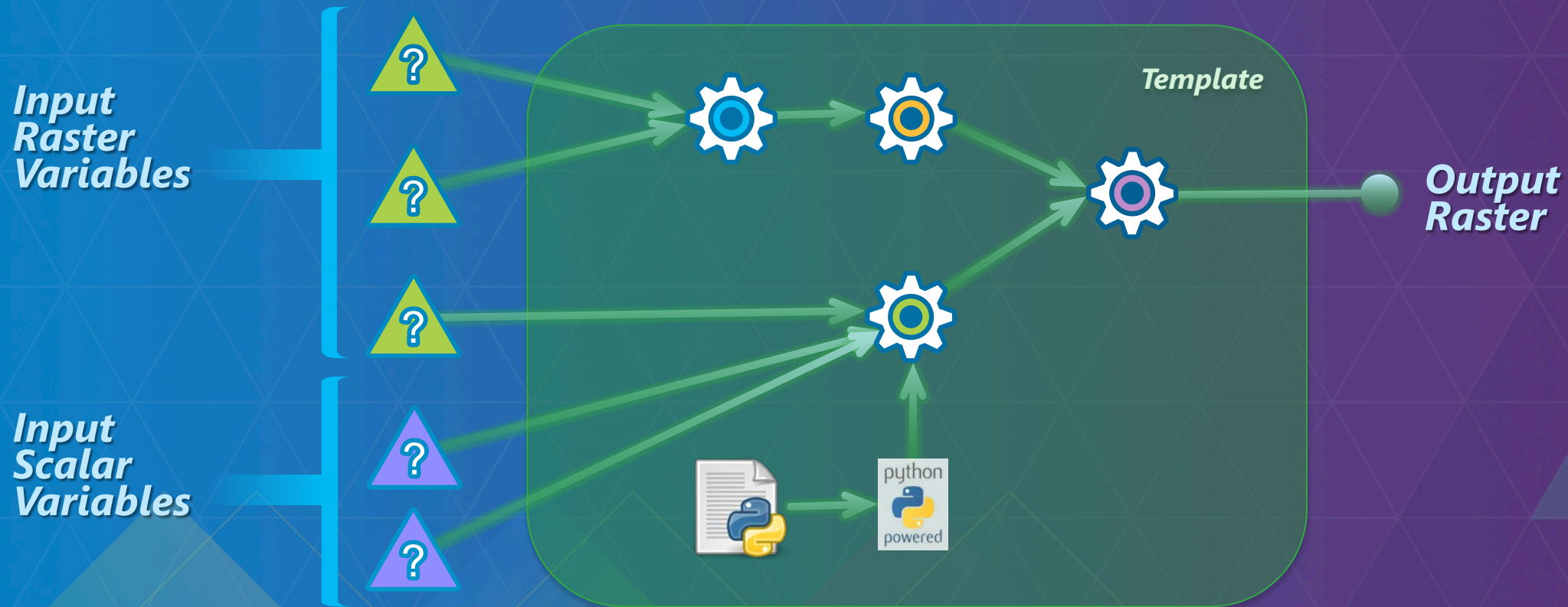


Raster Function *Templates*



What's a Raster Function *Template*?

The Basic Model



*A raster function template anonymizes select raster and scalar inputs as **Variables**.*

Raster Function Templates

Basic Workflows

- Create new function templates
 - Via *Raster Function Template Editor*
 - Layer > Symbology > Export As Raster Function Template
 - Function Raster Dataset > Functions > Save as
- *Function Raster Layer* in a Map
 - Image Analysis Window
 - Raster Functions Pane in *Pro*
 - Layer > Properties > Functions tab

*A raster function template makes your processing or analysis **portable**.*

Raster Function Templates

Advanced Workflows

- On a Mosaic Dataset
 - Populating a mosaic using the Add Rasters tool
 - Mosaic dataset items
 - *Batch Edit Raster Functions* or
 - *Edit Raster Function* Geoprocessing Tool.
 - *As Processing Templates*
- On an Image Service—for server-side processing

Learn more at <https://github.com/Esri/raster-functions#raster-function-templates>.

Raster Function Templates

Properties

- Name & Description
- Type: *Item, Group, or Mosaic.*
- Definition Query
- *Fields* that control *grouping.*

Raster Function Template Properties

Name: Windchill

Description: A raster function template that computes windchill on

Type: Item Group

Help:

Definition Query:

Thumbnail:

Group Items By:

Group Field Name: GroupName

Tag Field Name: Variable

OK Cancel

Demo

Processing Templates on a Mosaic Dataset



Demo

Publishing Processing Templates

Additional Considerations

Performance

- Vectorize.
- Use NumPy and SciPy.
- Use *Cython*: For production-grade performance.
- Bridge to C/C++ implementations via *ctypes* or *Boost.Python*.

Leverage well-known options to optimize time-critical parts of your raster function.

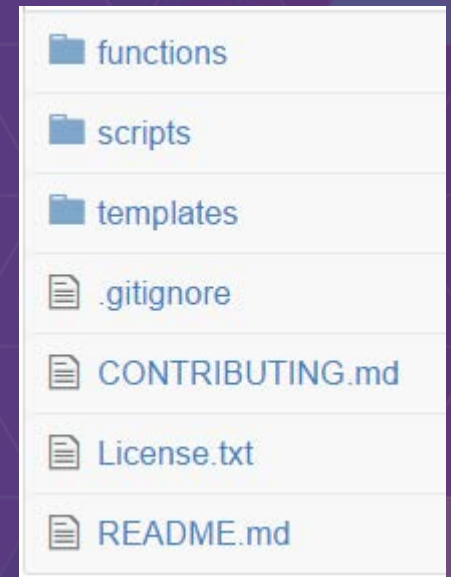
Publishing & Deployment

- Python version
 - Desktop vs. Pro: Python 2.7 vs. 3.4
 - Desktop vs. Server: 32- vs. 64-bit Python 2.7
- Package dependencies
- Binary deployment
 - CPython bytecode. Cython compiled binary.
 - *isLicensed* method to restrict usage.

GitHub

Where do functions and templates live?

- <https://github.com/Esri/raster-functions>
- Curated collection of raster *functions* and *templates*.
- Go ahead:
 - [Browse samples](#) to learn more.
 - [Fork](#) the repo. Experiment.
 - Log defects or enhancement requests as [issues](#).
 - Send [pull requests](#) to contribute code.



GitHub enables collaboration.

Wiki

Where do I find the story?

- <https://github.com/Esri/raster-functions/wiki>
- Details of interaction between ArcGIS and your Python raster function.
- Recommendations and techniques for writing effective raster functions

- **Anatomy of a Python Raster Function**

- getParameterInfo
- getConfiguration
- updateRasterInfo
- selectRasters
- updatePixels
- updateKeyMetadata
- isLicensed

- **Writing Effective Raster Functions**

- Performance Considerations
- Handling NoData
- Key Metadata
- Aggregation and Grouping
- Using Cython
- Debugging
- Deployment

The Road Ahead

- Tools for testing your raster functions—profiling or debugging.
- Streamline usability of Python raster functions in ArcGIS *Pro*.
- Integration with Portal workflows.
- Expand and maintain the *wiki*: API, recommendations, [resources](#).

...for Pythonistas and Pythoneers with a passion for scientific computing.

In Closing

Wake up! We are done.

- Powerful pipeline for processing, analysis, and visualization.
 - *Raster Functions. Chains. Templates.*
- Using simple constructs in Python: participate and exploit.
- Code and docs are on [GitHub](#).
- We'll help you along the way—[@jdrisdelle](#) and [@akferoz](#).

Rate This Session

www.esri.com/RateMyDevSummitSession

Sessions on related topics:

Python: Working with Raster Data—Wednesday at 4:00

Python: Working with Scientific Data—Wednesday at 5:30

<http://www.esri.com/events/devsummit/agenda>