Writing Image Processing Algorithms using the Python Raster Function

Jamie Drisdelle

Agenda

- Introduction
- Anatomy of a raster function
- Building raster models
- Additional considerations
- Q&A

Introduction

Raster Functions

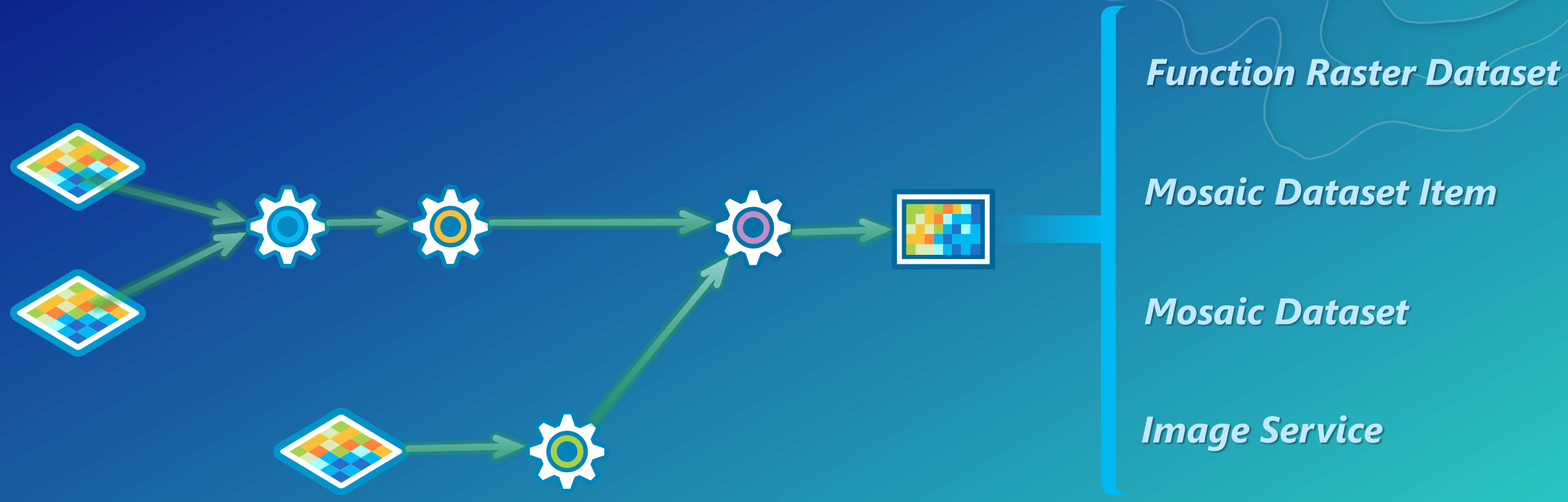
What’s a Raster Function?

- Mapping of one raster to another.
- On-demand. Transient.
- Different from a geoprocessing tool.

- … a transformation of one raster into another.

Chaining Raster Functions

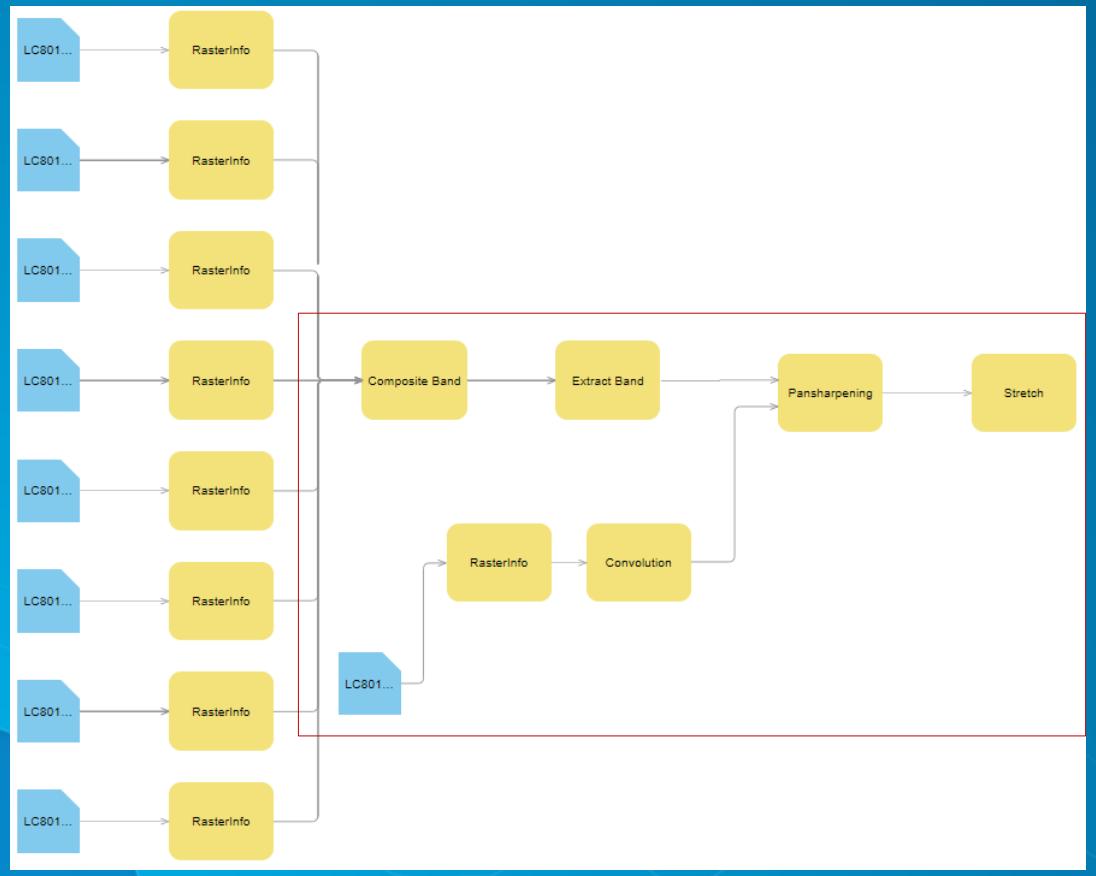
- A raster model encapsulates your algorithm as a tree of functions.

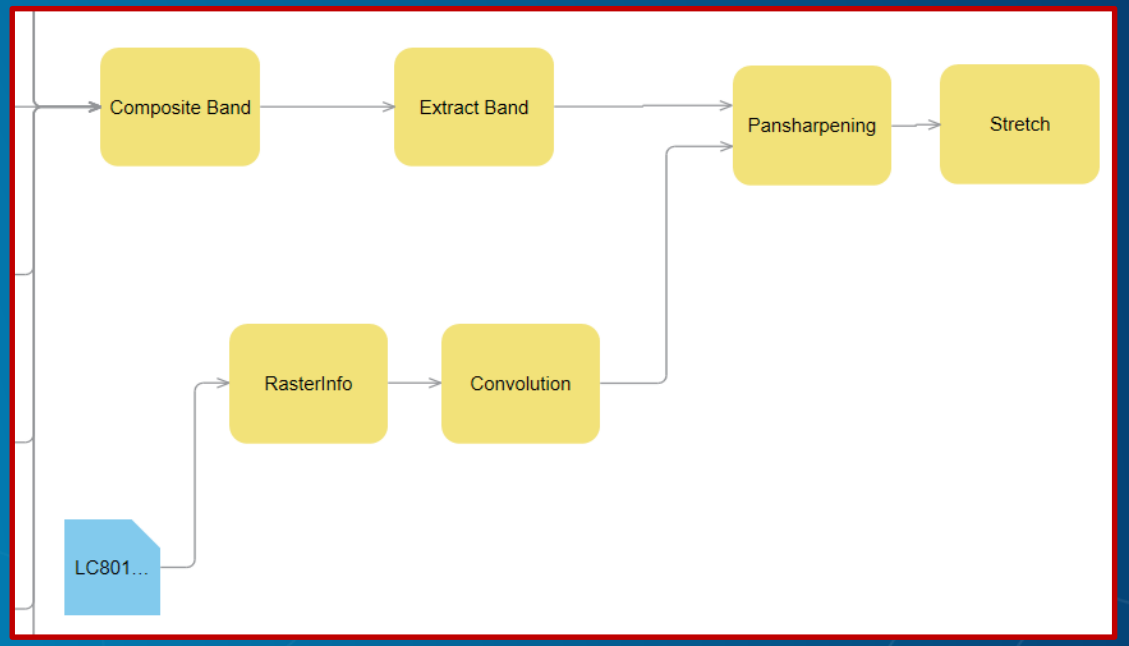


- Raster Model

Chaining Raster Functions

- **An example: Landsat 8 Pansharpened**





Demo

- Apply a **raster functions** to image layers in ArcMap

  Browsing **raster models** on an image service on Portal

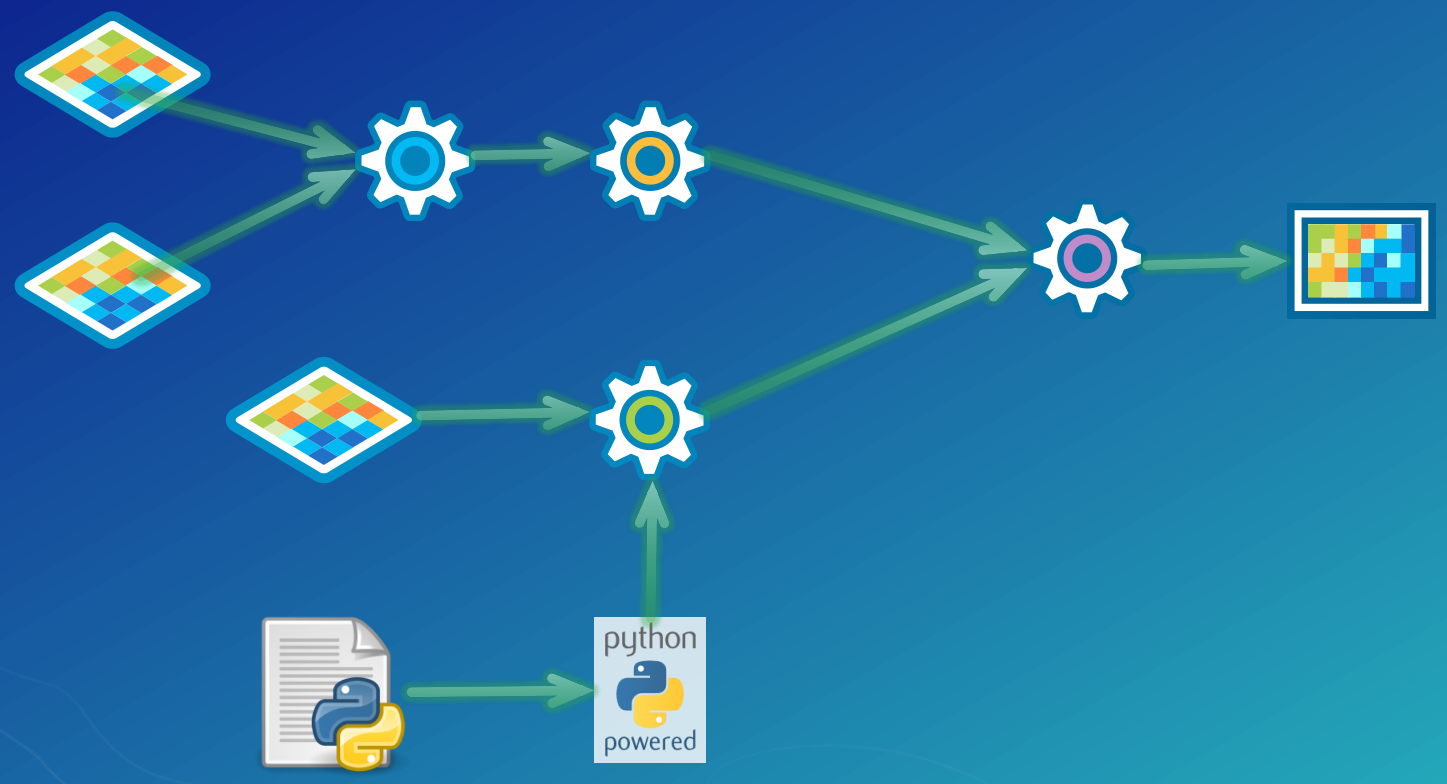
Python Raster Functions

What's a *Python* Raster Function?

- Transforming rasters—image processing and analytic algorithms—in Python.
- 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**



Function Raster Dataset

Mosaic Dataset Item

Mosaic Dataset

Image Service

Motivation

- Extend ArcGIS—participate in a *raster model.*
- Primary pipeline for image data in ArcGIS—*processing*, *analyzing*, and *visualizing*.
- On-the-fly at display resolution, at full resolution using distributed raster analysis.
- 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.”

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!

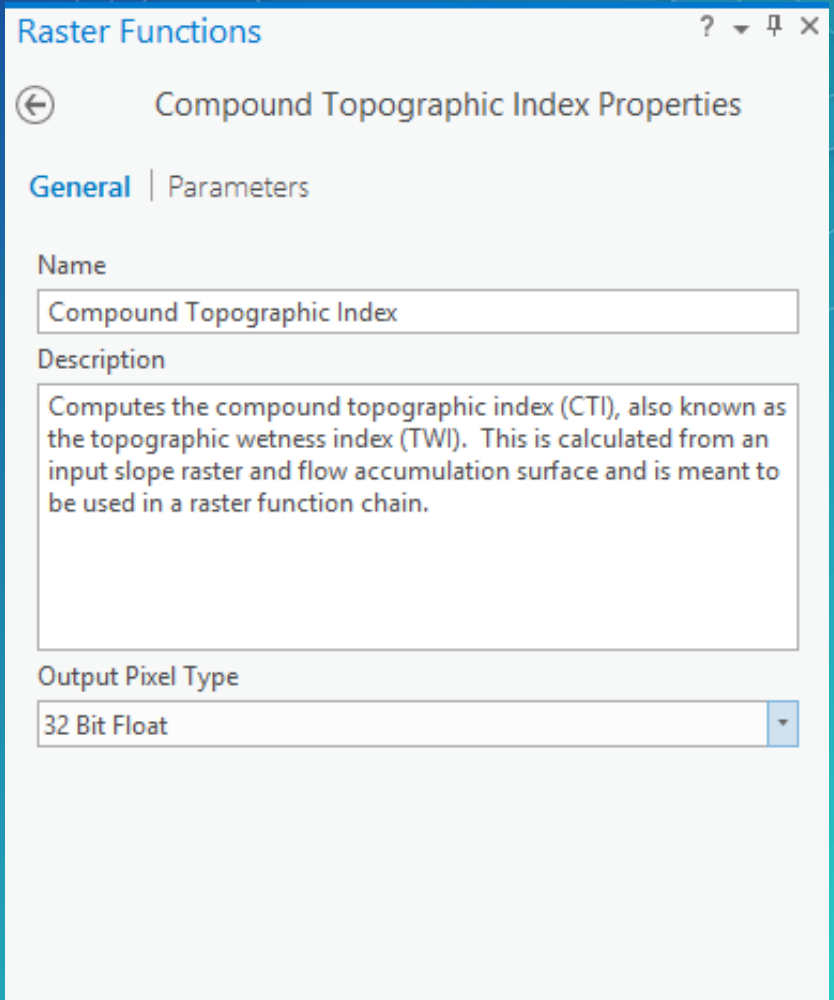
```python
import numpy as np

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.*

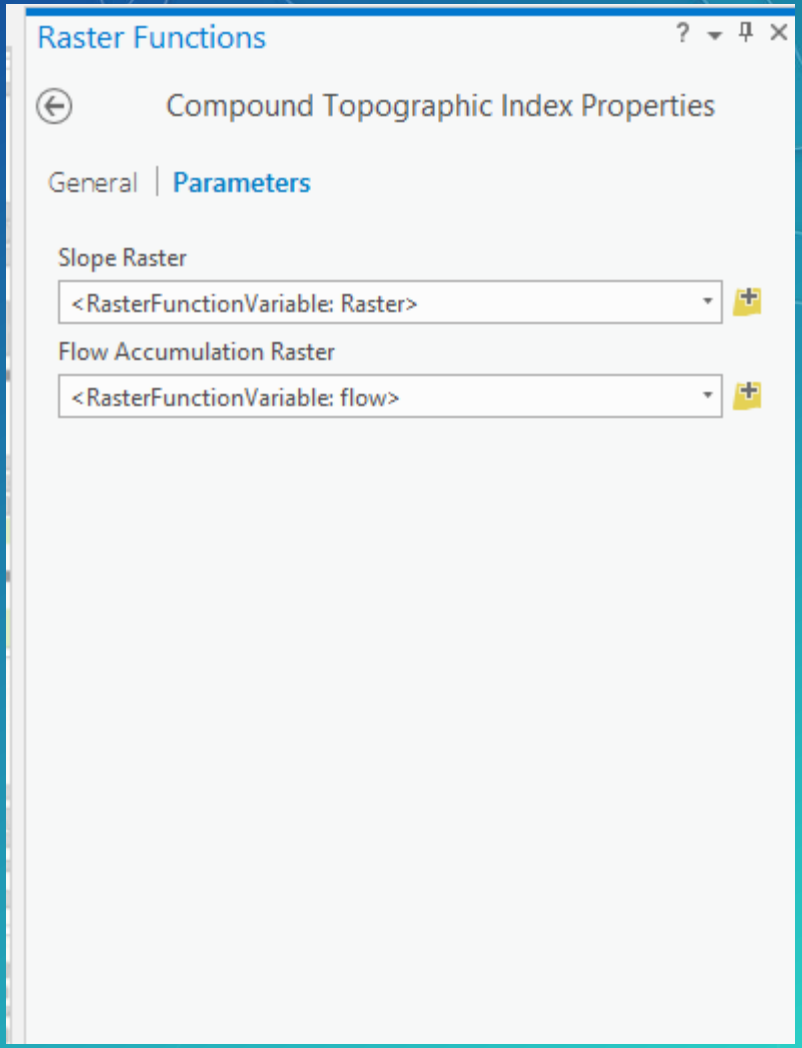


The API

- `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



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**

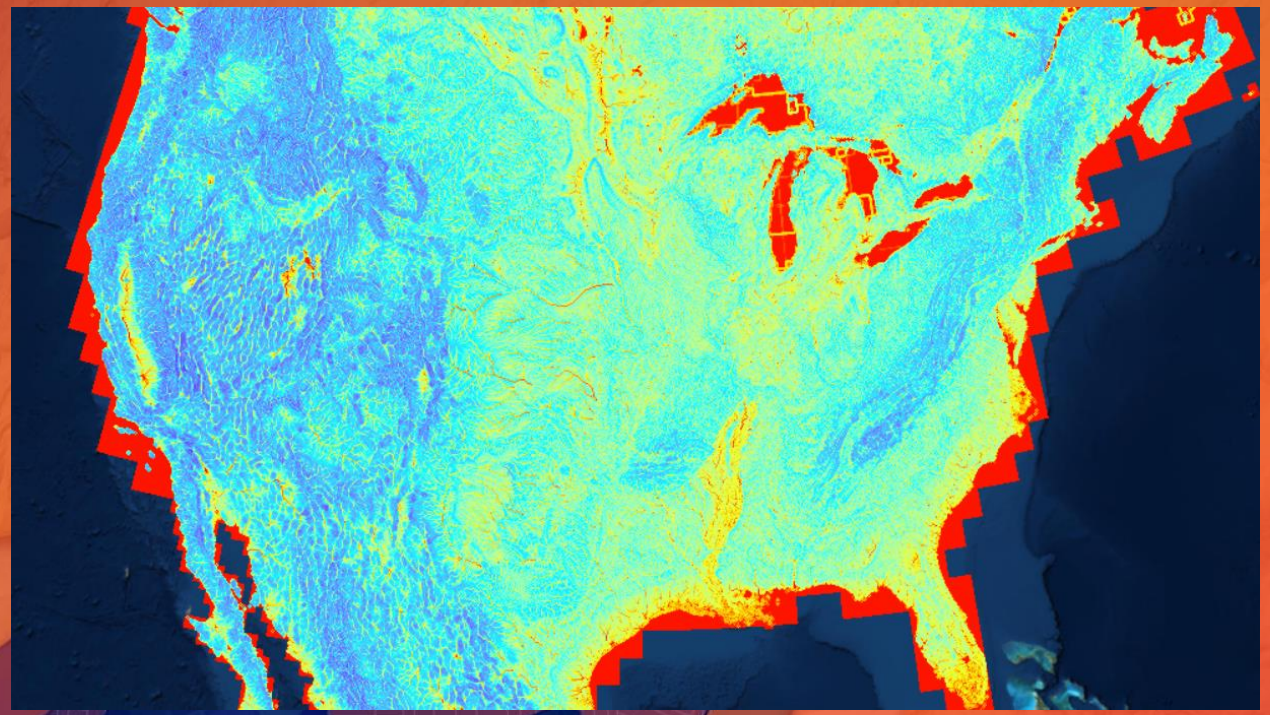
- ***Returns:*** **updated values of given properties**

The API

- `isLicensed()`

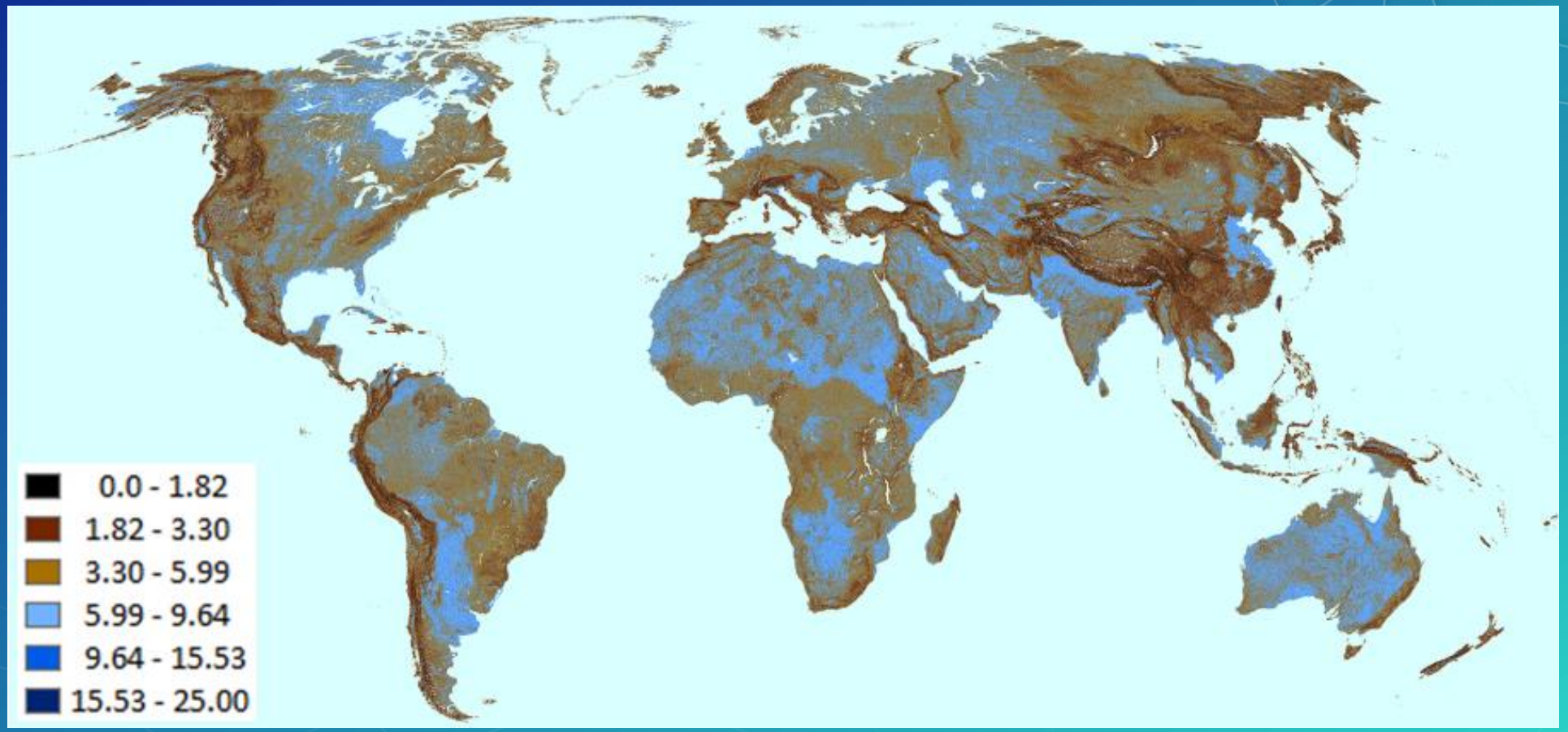
  - ***Given:***
    - **info on parent product,**
    - **context of execution.**
  - ***Returns:***
    - **OK to Run (Boolean)—Licensed to execute or not?**
    - **Expected product level and extension.**

Compound Topographic Index



Compound Topographic Index (CTI)

- A steady state wetness index, a.k.a. Topographic Wetness index (TWI)



T.R. Matthews et al. 2015

Compound Topographic Index (CTI)

- Used to study spatial scale effects on hydrological processes and to identify hydrological flow paths for geochemical modelling.
- Used to characterize biological processes such as annual production, vegetation patterns, and forest site quality.

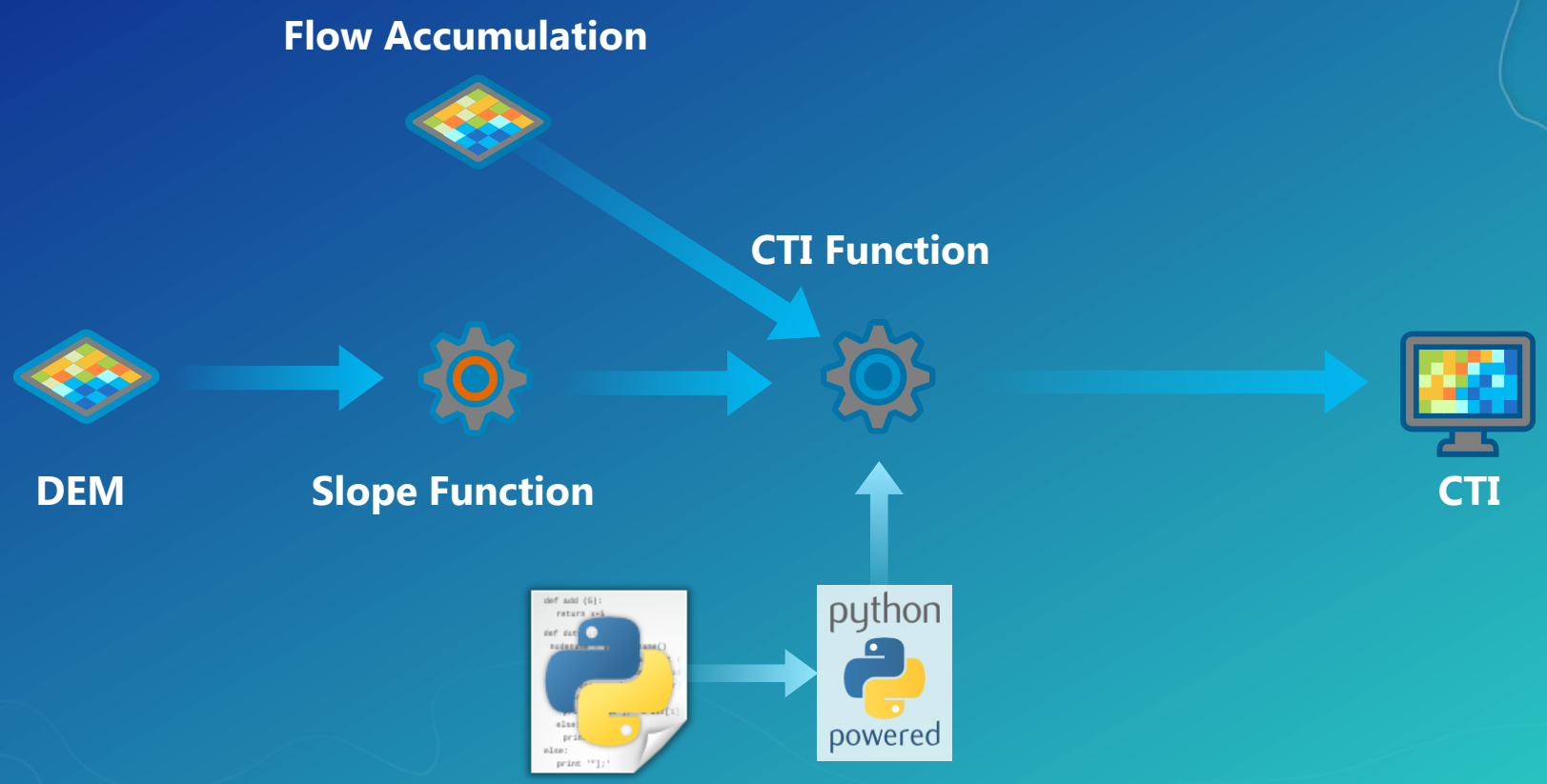
Compound Topographic Index

- **Calculations**

$$CTI = \ln\left((flow_{accumulation} + 1) * \frac{cellsize}{\tan(slope)}\right)$$

Compound Topographic Index

- Calculations



CTI: **Code**

```
calc_cti(slope, flow_acc, cellsize):
tan_slope = np.tan(slope)
tan_slope[tan_slope==0]=0.0001
cti = np.log(((flow_acc+1)*cellsize)/tan_slope)
return cti
```

Demo

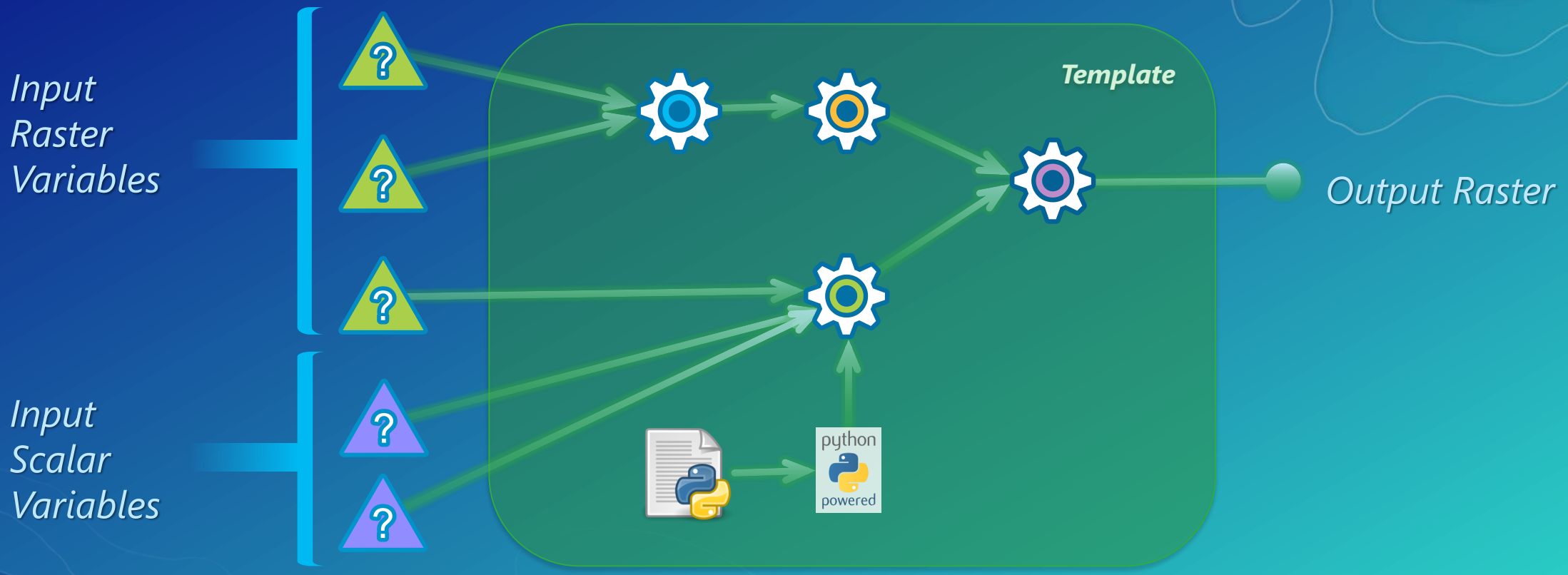
- Apply ***CTI*** raster model to image layers on map using ArcGIS Pro

Building and Using *Raster Function Templates*

- Templatized chains of raster functions

What's a Raster Function Template?

- **The basic concept**



- **A templatized raster model replaces one or more inputs with place-holder Variables.**

Raster Function Templates

- Basic workflows with “function templates”

- Create new function templates
  - Via *Raster Function Template Editor*
  - Layer > Symbology > Export As Raster Function Template
  - Function Raster Dataset > Functions > Save as
  - Symbology as a template

- *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 with function templates

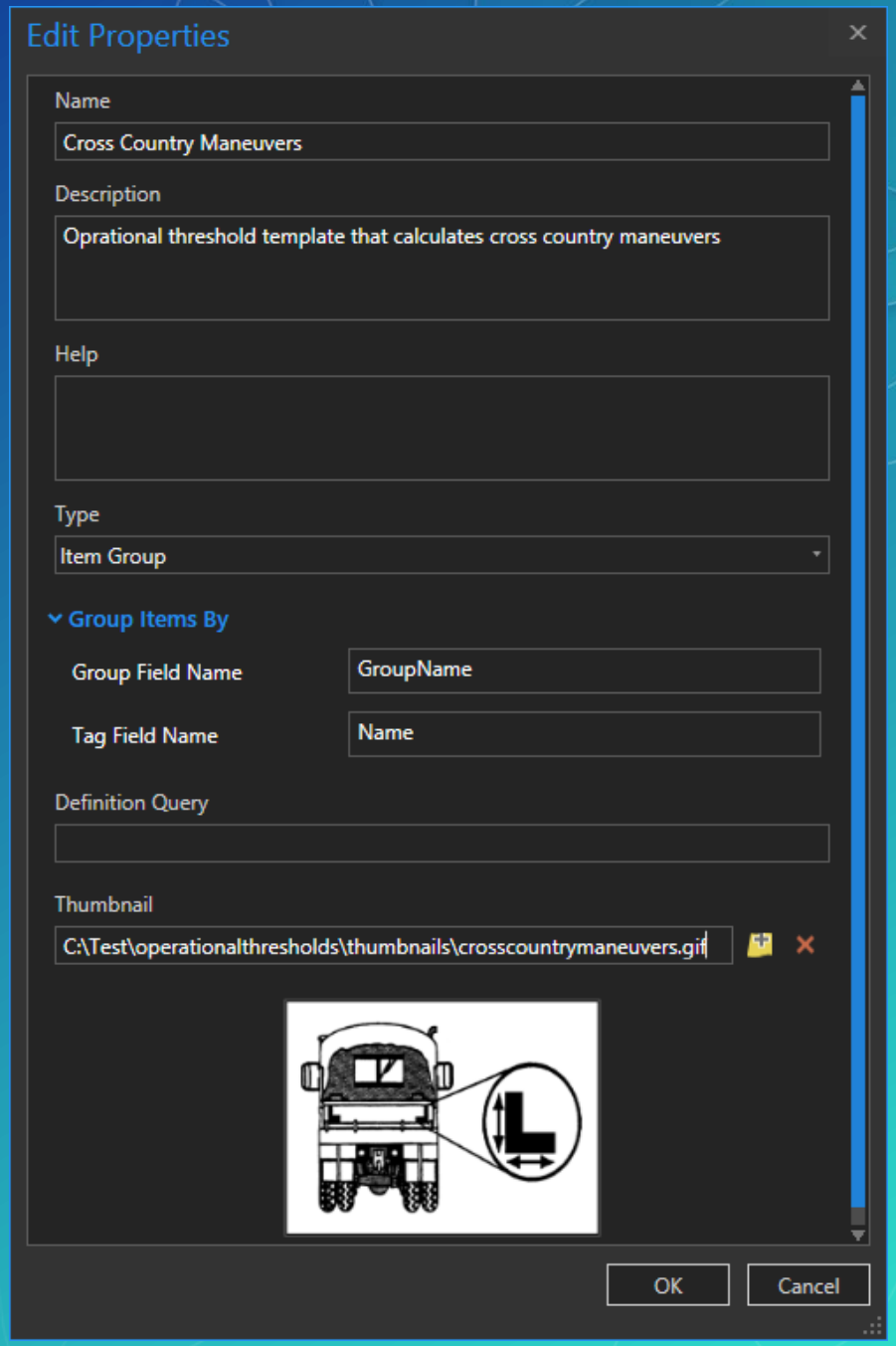
- On a Mosaic Dataset
  - Populating a mosaic using the Add Rasters tool
  - Mosaic dataset items
    - *Batch Edit Raster Functions or*
    - *Edit Raster Function* Geoprocessing Tool.
  - As *Processing Templates*

- On an Image Service—for server-side processing

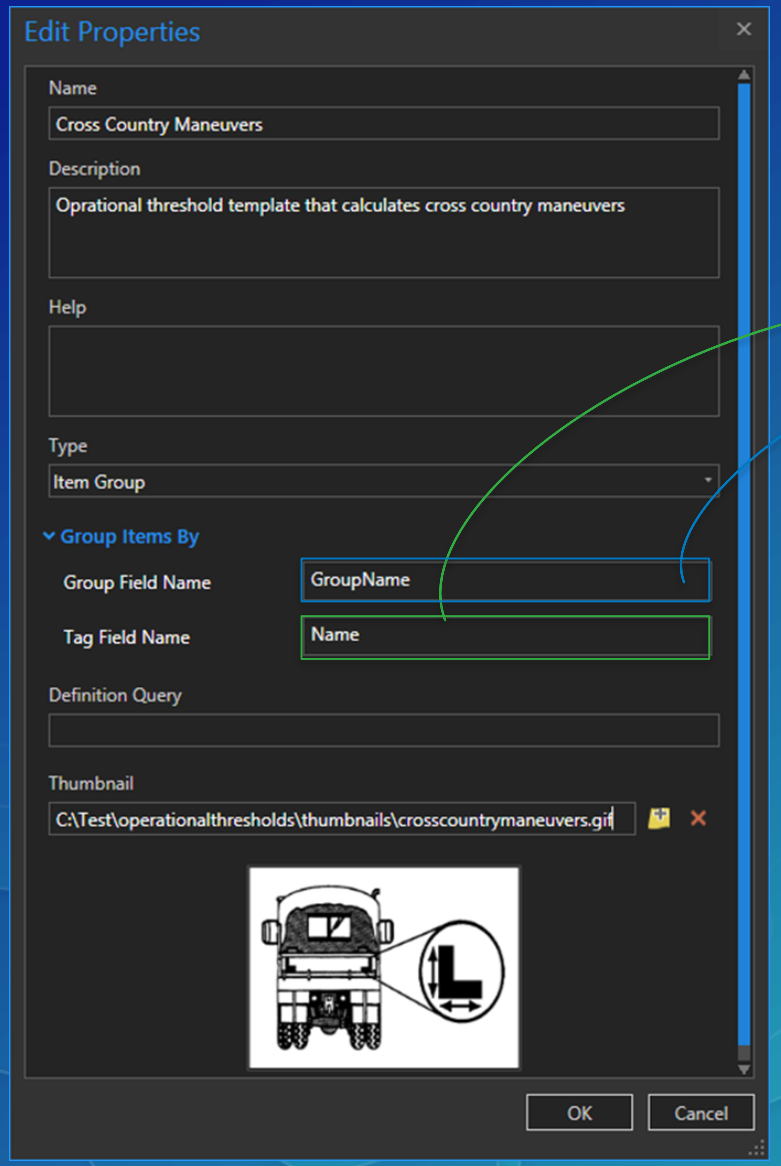
- Learn more at https://github.com/Esri/raster-functions#raster-function-templates.

Raster Function Templates

- **Properties of a function template**
- **Name & Description**
- **Type: *Item*, *Group*, or *Mosaic.***
- **Definition Query**
- ***Fields* that control *grouping*.**



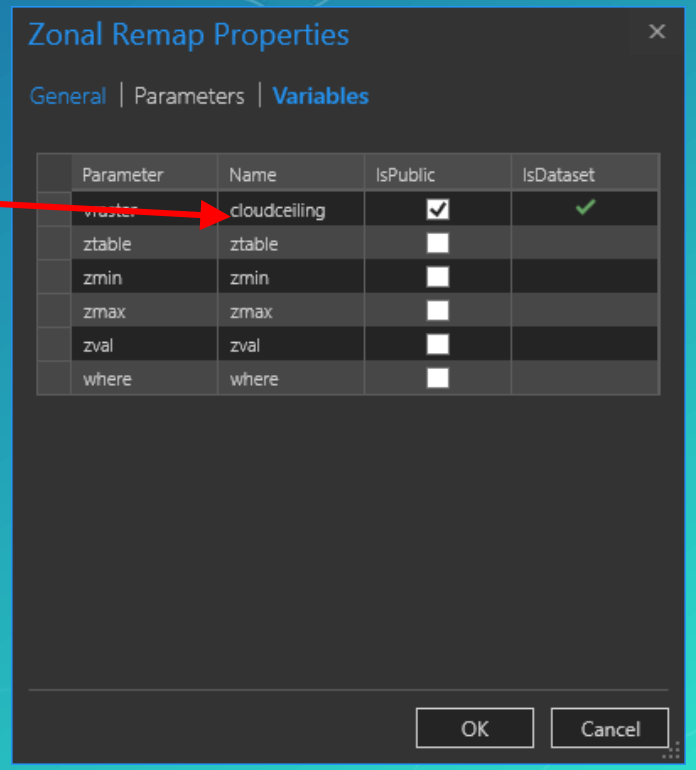
Using Mosaic Dataset Items in a Mosaic Dataset Template



OBJECTID	Raster	Name	GroupName
8	Raster	cloudceiling	weather
9	Raster	heatindex	weather
10	Raster	snowdepth	weather
11	Raster	temperature	weather
12	Raster	visibility	weather
13	Raster	windchill	weather
14	Raster	windspeed	weather

Group and tag fields in template properties are associated with fields in the mosaic dataset table

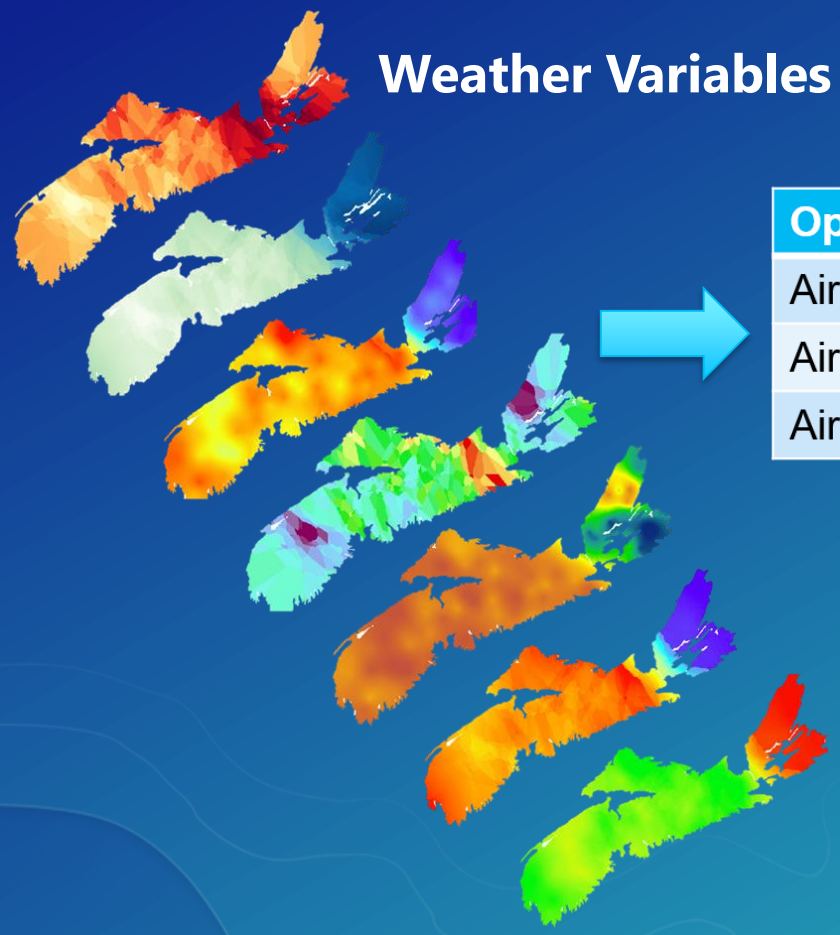
Name being passed into the function will be the record value



Site Suitability Analysis

Site Suitability Analysis

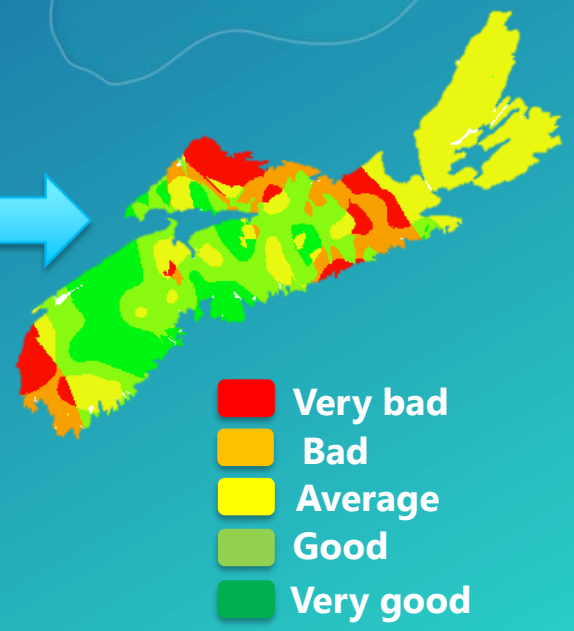
Weather Variables



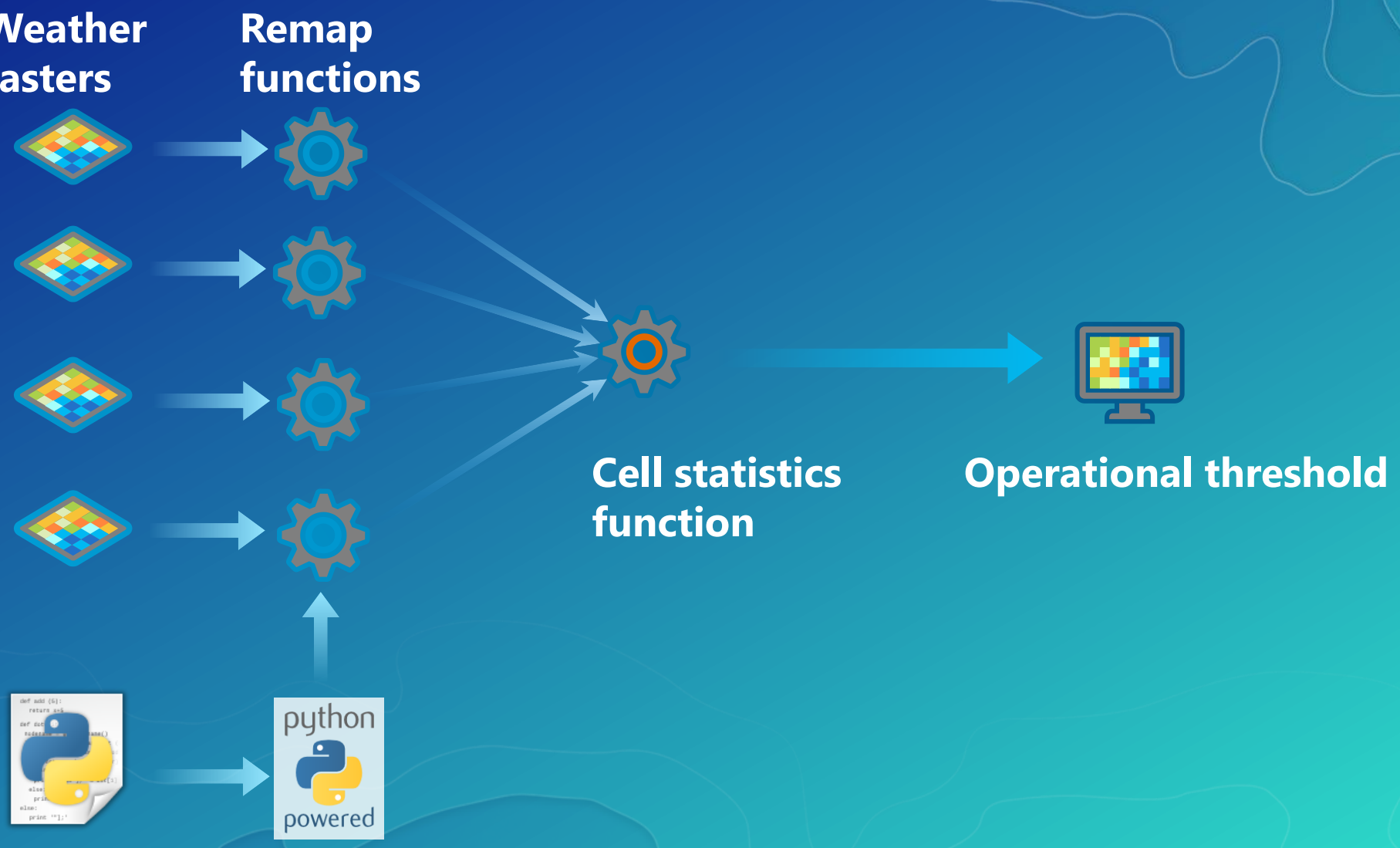
Remap Table

Operation	Variable	From	To	Remap
Air Defense	Temperature	50	100	-2
Air Defense	Cloud Ceiling	2460	4920	-1
Air Defense	Wind speed	0	36	0

Output Function Raster



Site Suitability Analysis



Site Suitability Analysis

```
# use zonal thresholds to update output pixels...
if ZT is not None and len(ZT.keys()):
    for k in (zoneIds if zoneIds is not None else [None]):
        T = ZT.get(k, None)         # k from z might not be in ztMap
        if not T:
            continue

        for t in T:
            I = (z == k) if z is not None else np.ones(v.shape, dtype=bool)
            if t[0] and t[1]:       # min and max are both available
                I = I & (v > t[0]) & (v < t[1])
            elif t[0]:
                I = I & (v > t[0])
            elif t[1]:
                I = I & (v < t[1])
            p[I] = (t[2] if t[2] is not None else self.defaultTarget)

pixelBlocks['output_pixels'] = p
```

Demo

- Apply ***suitability*** r aster model to a mosaic dataset using ArcMap
- Browse processing templates of the mosaic layer

Additional Considerations

Performance

- Do not use arcpy or array looping
- Use NumPy and SciPy.
- No significant improvement performance when using Cython.

- 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.**

Demo

- Automatic subcategory generation in ArcGIS Pro

GitHub

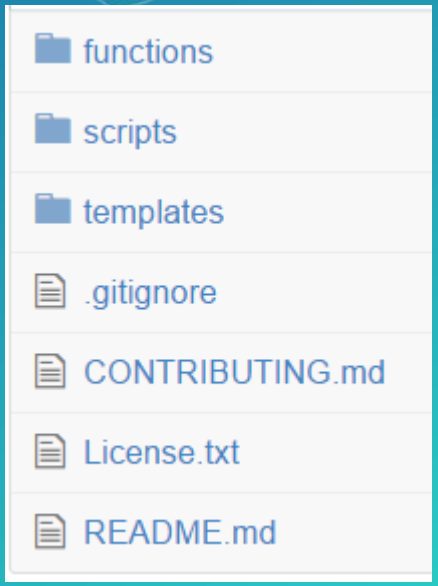
- Where do functions and templates live?

- 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**

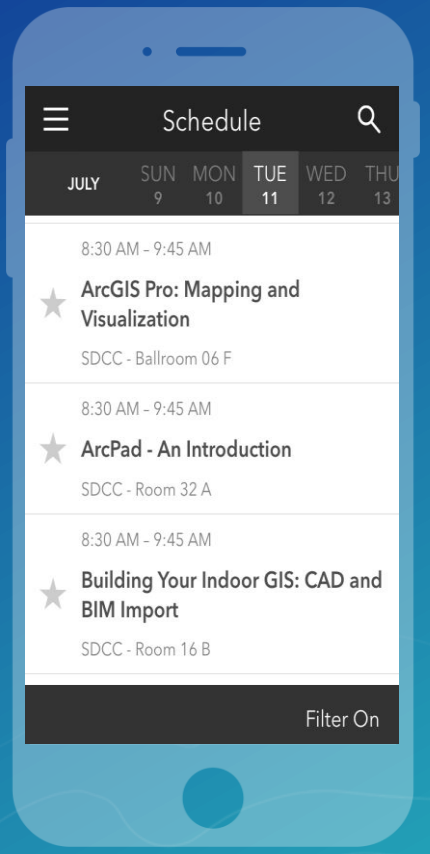
We'll help you along the way @jdrisdelle, and **@Zoeee1616** .

- **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

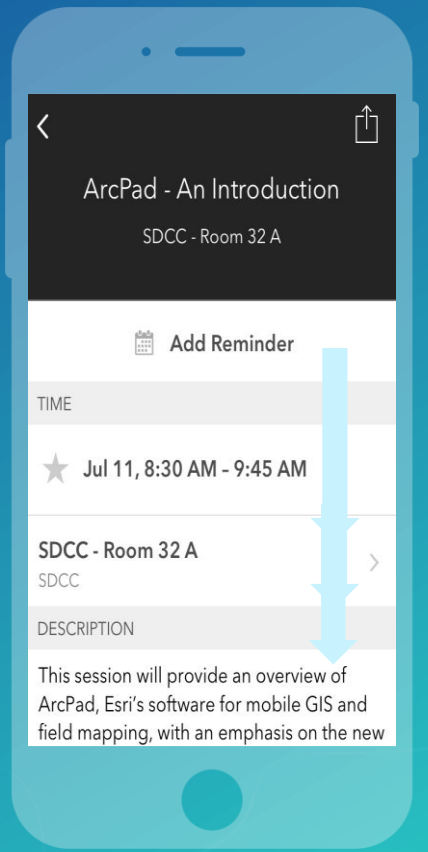
Please Take Our Survey on the **Esri Events App!**

Download the Esri Events app and find your event

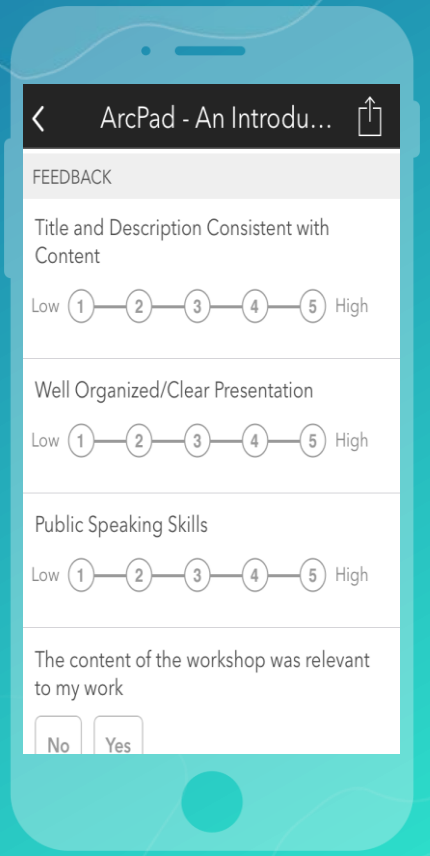
Select the session you attended



Scroll down to find the survey



Complete Answers and Select "Submit"



esri

THE SCIENCE OF WHERE