PARIS LODRON UNIVERSITÄT SALZBURG
Fachbereich Geoinformatik

Practical Implementation of Map Algebra

Yangyu Li | yangyu.li@stud.plus.ac.at

Agenda

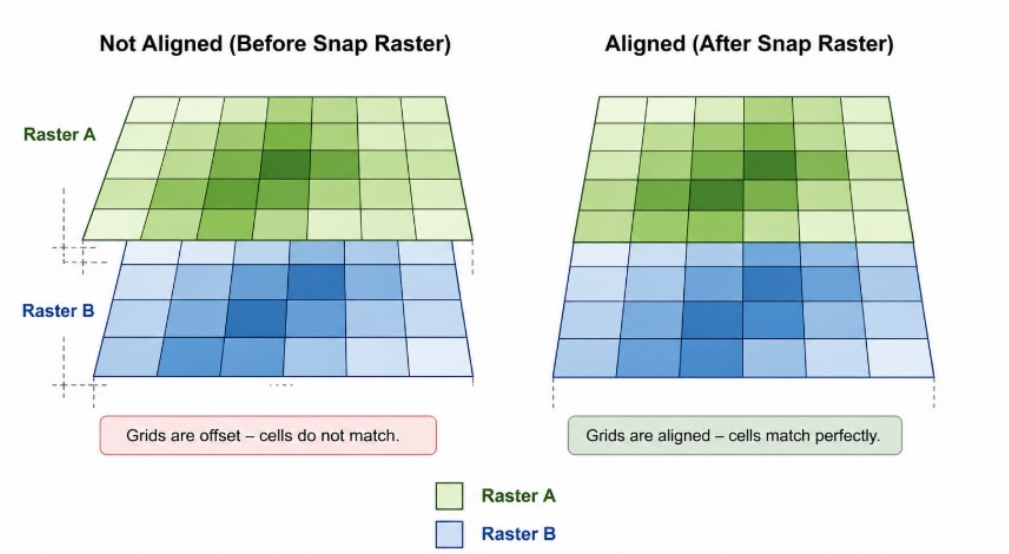
1. Scope
2. Environments setting
3. Expression-based implementation
4. Tool-based implementation
5. Workflow-based implementation
6. Script-based implementation
7. Example: ModelBuilder vs. ArcPy
8. Strength and limitations
9. Why open-source implementation matters
10. GEE case study
11. Conclusion

Scope

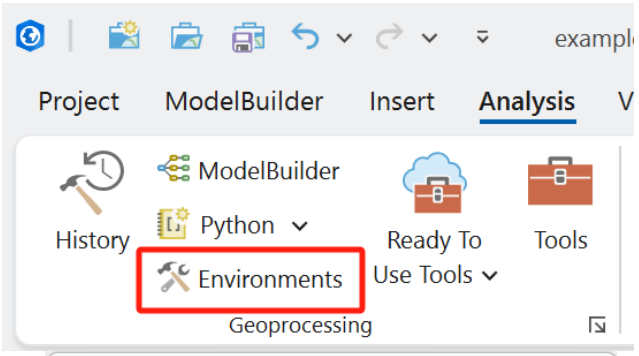
- Focus on implementation, not on application domains
- How Map Algebra is implemented in different environments
- Mainly Focus:
  - Spatial analysis tools (ArcGIS Pro)
  - ModelBuilder (ArcGIS Pro)
  - ArcPy (ArcGIS Pro)
  - QGIS
  - GRASS GIS
  - Open-source Python packages

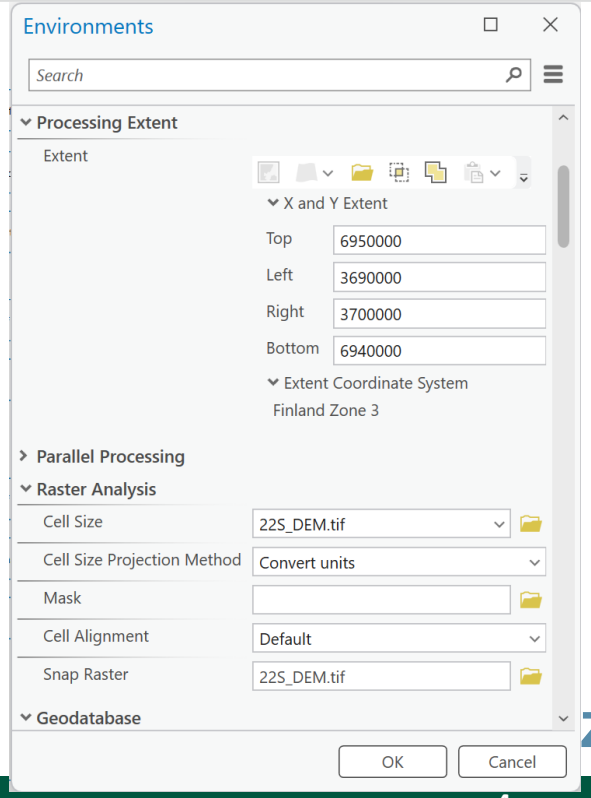
Environment settings

- Cell size
- Extent
- Snap raster
- Mask
- NoData propagation
- Edge handling
- …



Generated by ChatGPT





Main implementation modes

- Expression-based implementation
  - Direct raster calculation expressions
- Tool-based implementation (Spatial analysis tools)
  - Focal statistics
  - Reclassify
  - Zonal statistics
  - Distance tools
  - …

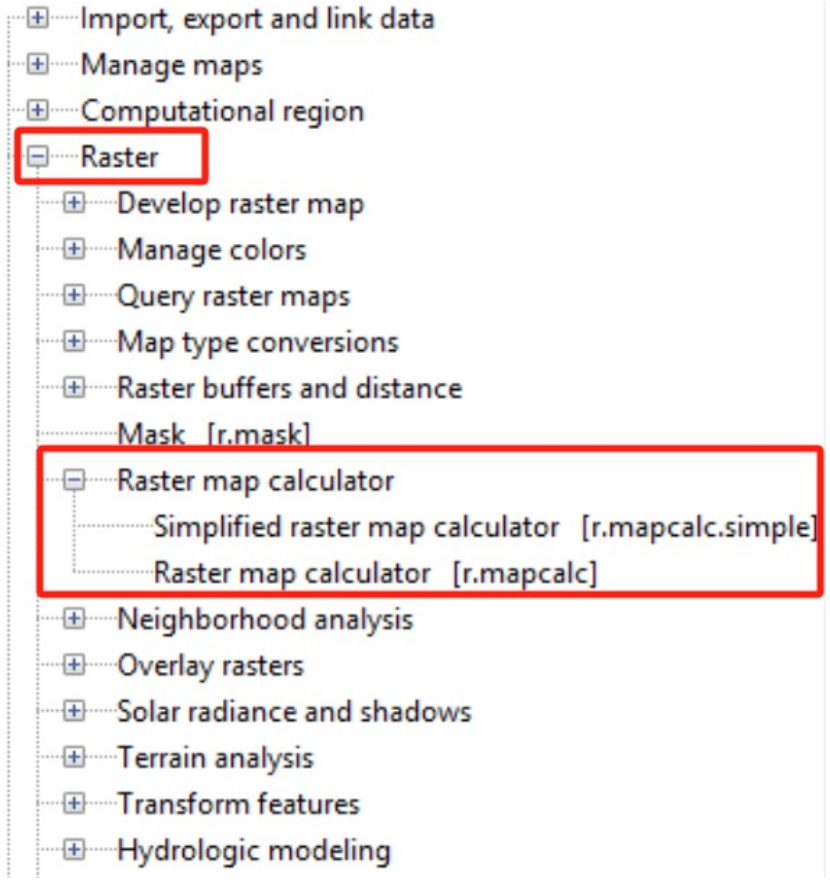
Main Implementation modes

- Workflow-based implementation
  - ModelBuilder
  - Chained geoprocessing tools
- Script-based implementation
  - ArcPy
  - PyQGIS
  - Python expressions and automated workflows

Expression-based implementation

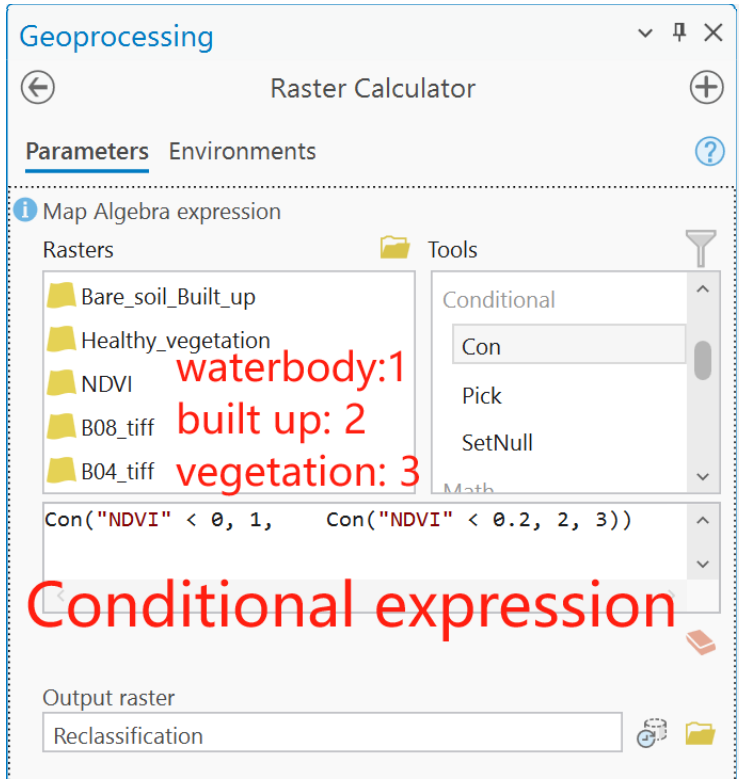
- Direct map expressions
  - ArcGIS Pro: Raster Calculator
  - QGIS: Raster Calculator
  - GRASS GIS: Raster map Calculator
  - → Raster layers are written as variables in expression

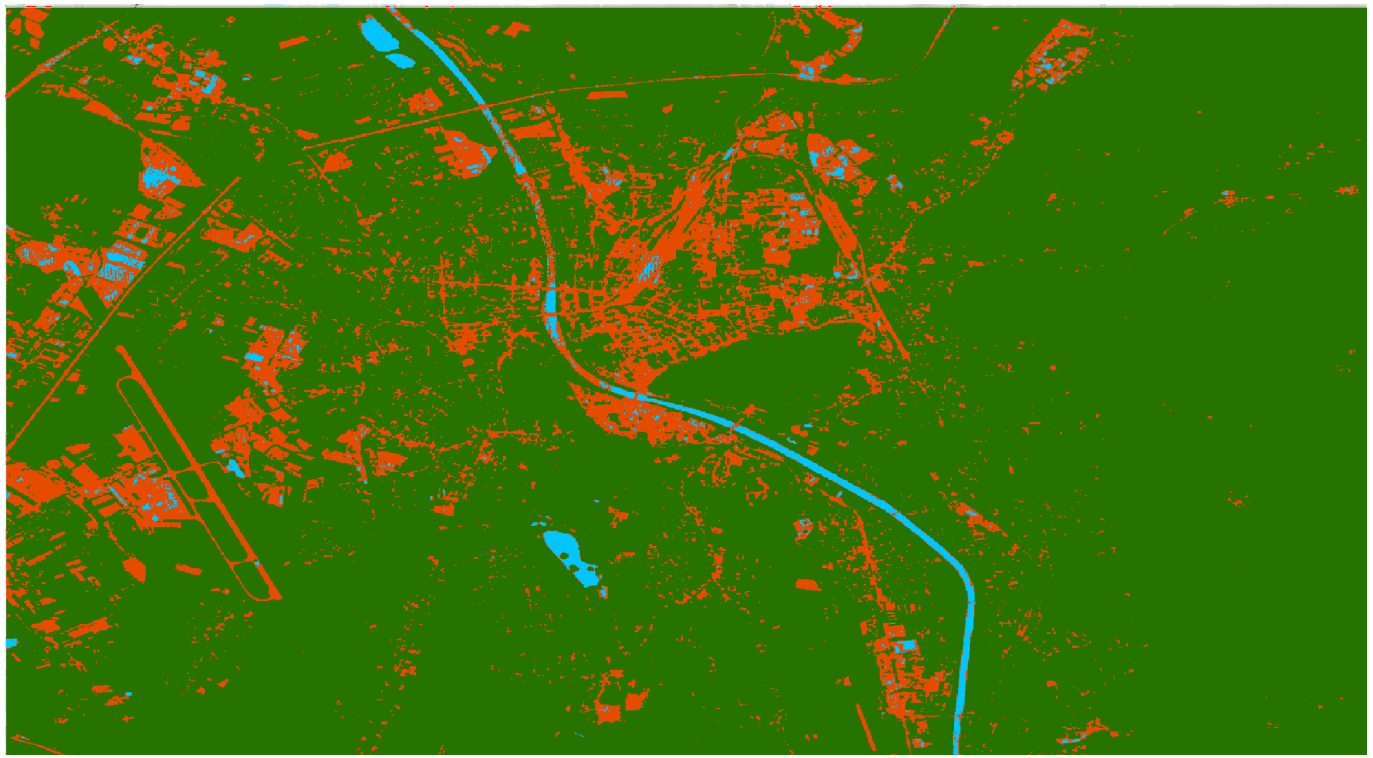
- Supports:
  - Mathematical operations
  - Logical expression
  - Boolean expression
  - Conditional expression



Expression-based implementation

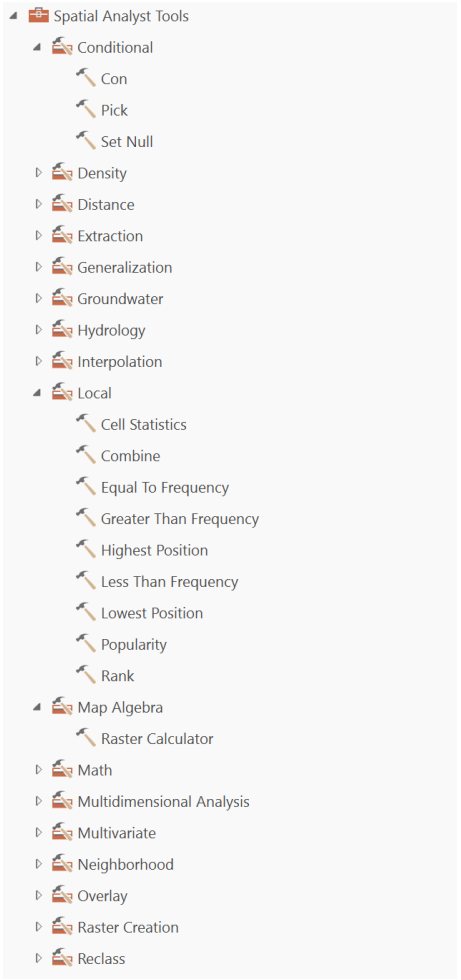
❖Exmaple:

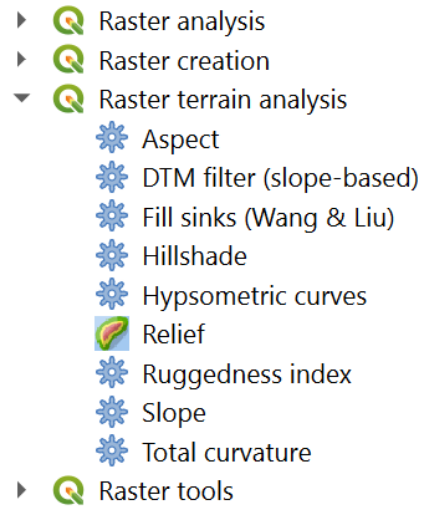


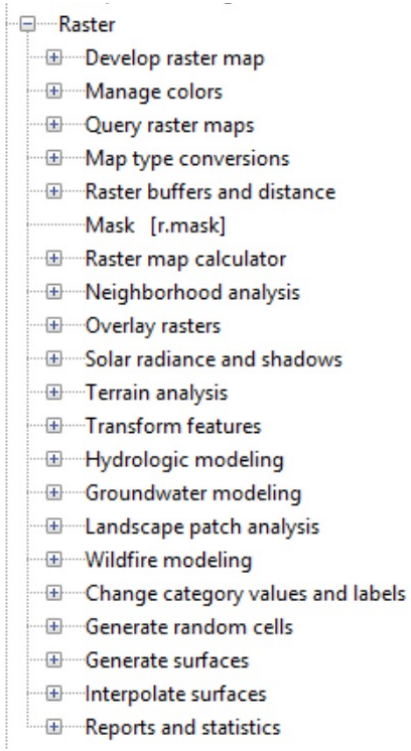


Pre-structured implementation (tool-based implementation)

- ArcGIS Pro:
  - Reclassify
  - Cell statistics
  - Focal statistics
  - Zonal statistics
  - Slope
  - Aspect
  - …
- Open-source examples:
  - QGIS Processing tools
  - GRASS raster modules

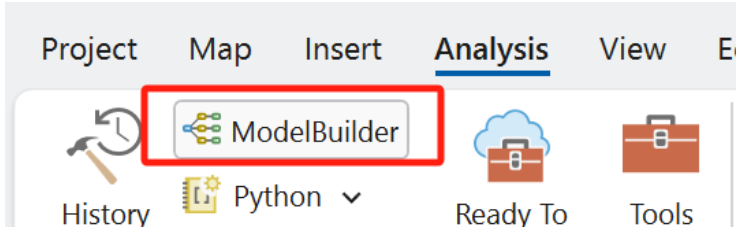


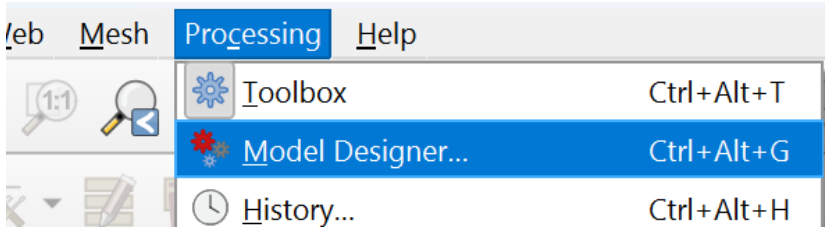


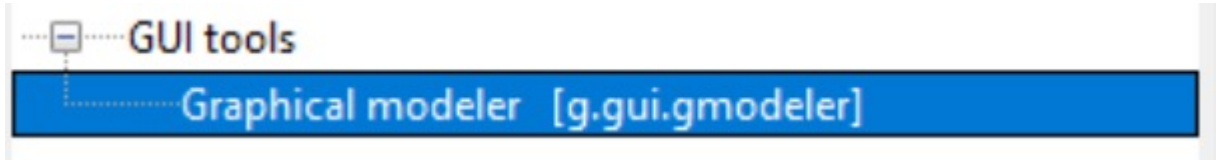


Workflow-based implementation

- Multiple raster operations can be chained into structured workflows
- ArcGIS Pro: ModelBuilder
- QGIS: Model Designer
- GRASS: Graphical Modeler



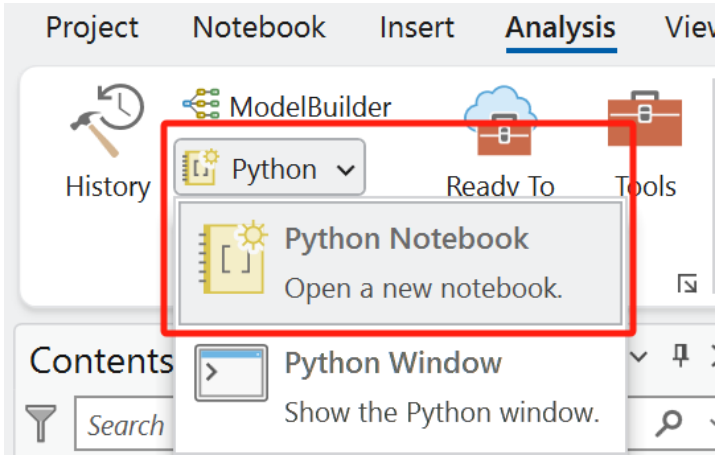




- Useful for:
  - Building structured analytical workflows
  - Making raster analysis reproducible
  - Suitable for non-programmers

Script-based implementation

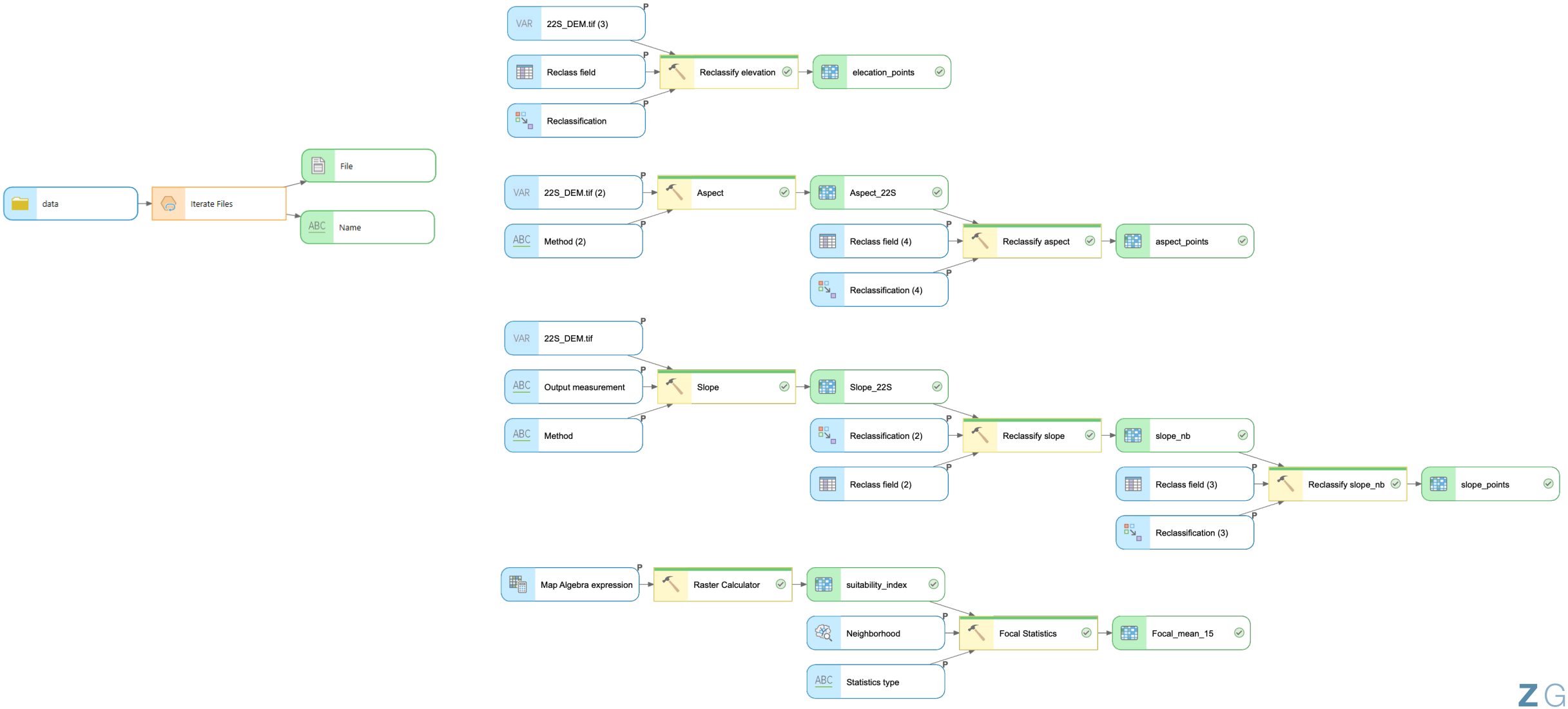
- move raster operations into code
- Supports:
  - Loops
  - Batch processing
  - Automation of repetitive analyses
  - Parameter control
  - Reproducibility
- ArcPy: ArcGIS Pro's API
- Open-source Python:
  - PyQGIS
  - GDAL-based scripting
  - Numpy
  - Rasterio
  - Rioxarray / xarray



C: > Users > Liyan > Desktop > demo.py

```
1  import numpy as np
2  import rasterio
3  import xarray as xr
4  import rioxarray as rxr
5
```

Example: ModelBuilder vs. ArcPy



Example: ModelBuilder vs. ArcPy

Map | ArcPy_example

Code

Import packages

[113]:
```
import arcpy
from arcpy.sa import *
import os
```

Add result to Map

Add an output to the current map

[114]:
```
def add_to_current_map(data_path):
    try:
        # If the script is run inside ArcGIS Pro, add output to map
        aprx = arcpy.mp.ArcGISProject("CURRENT")
        m = aprx.activeMap
        if m:
            m.addDataFromPath(data_path)
    except Exception:
        # If the script is run outside ArcGIS Pro, just skip adding to map
        pass
```

Workflows

[123]:
```
def workflows(dem, gdb):
```

Workflow-based vs. Script-based

- **Workflow-based implementation**
- Strength:
  - **Visual and intuitive** workflow design
  - Integrates directly with ArcGIS geoprocessing tools
  - Easy to understand for **non-programmers**
  - Good for **demonstrating** analytical logic
  - **Convenient for relatively simple workflows**
- Limitations:
  - **Less flexible** for complex conditions and custom logic
  - Batch processing is possible, but often requires iterators and careful variable setup
  - **Output naming and file management**
  - Large models may become difficult to maintain and debug

Workflow-based vs. Script-based

- **Script-based implementation**
- Strength:
  - **Highly flexible** for automation and batch processing
  - Can **iterate** through folders and datasets **automatically**
  - **Output names** can be generated **dynamically** to **avoid overwriting**
  - Easier to include **conditional logical, loops, and error handling**
  - Better suited for reproducible and scalable workflows
- Limitations:
  - Requires **programming knowledge**
  - **Not intuitive** compared to a visual model
  - Harder for beginners to learn and debug
  - Scripts need more management of parameters and environments

Why open-source implementation matters

- No license barriers
- Accessible for teaching and research
- Transparent and reproducible
- Map Algebra is not tied to one software vendor
- Supports portability and broader adoption

Thank you for your attention!