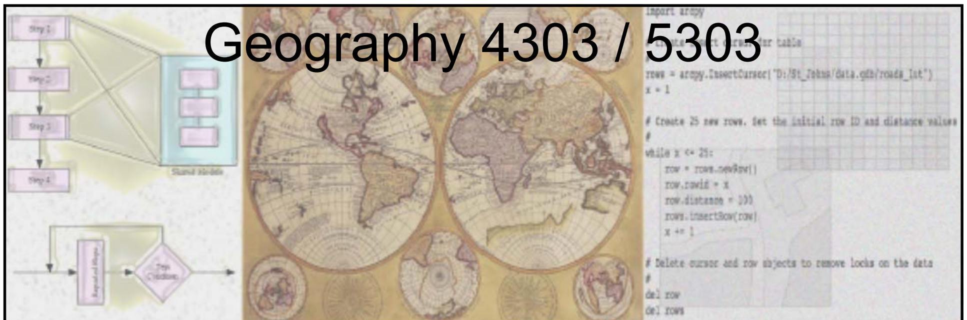


GIS PROGRAMMING FOR SPATIAL ANALYSIS

Class 08: Raster Overlay, Map Algebra, Neighborhood Functions

Some Updates

- Lab 5 work and submission
- Groups for final projects
- Schedule for project work to be discussed later

Last Lecture / Last Week

- Raster data structure and properties
- Raster objects (arcpy) and how to use them
- Intro to Numpy (multiarray object, Ufuncs, convenience funcs)
- Some first examples on how to use numpy's array objects

Today ‘s Outline

- Learn how to get **full access** to an image and how to **divorce processing** from **data structure**
- Understand how to **develop** functionality for raster analysis
- Learn how to carry out raster overlay and map algebra by using numpy arrays as 2D **matrices**:
  – **Local** (pixel-by-pixel) raster functions
  – **Focal** (neighborhood) raster functions
- Exercise: Write your own Mean Filter

Learning Objectives

- Principles of **programming** for raster analysis
- Getting **access to raster data objects**
- How to use **Numpy's** 2D **array structures** for raster analysis
- How to write your own map algebra as local and focal functions

Critical to raster analysis…

Part I: Data access: Read, Write raster objects

Part II: Developing local Map Algebra functions

Part III: Developing neighborhood (e.g., focal) Map Algebra functions

Part I: Data access
Read, Write raster objects

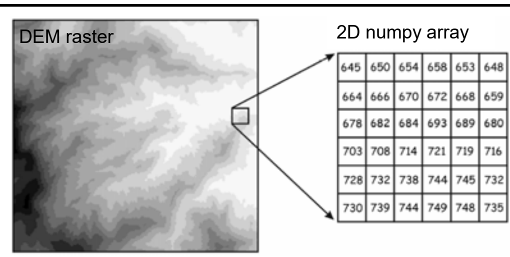
In general, to carry out any form of raster analysis, you need:

(1) **Access** to the image contents (**read** the image)

(2) Use the 2D array to develop raster analysis functions (see Part II)

(3) **Write** the results into a **new raster** dataset

Access the raster dataset



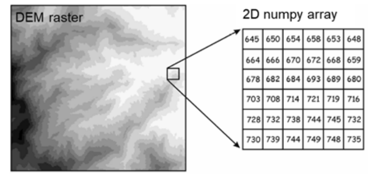
- **Load** an image object
- **Read** single **bands** (multi-spectral remote sensing images)
- **Retrieve properties** (projection, cell size, data type)
- Different tools/ approaches:
  - Using arcpy/ numpy functionality
  - ASCII grids and numpy functionality
  - "Geospatial Data Abstraction Library GDAL" : https://pypi.python.org/pypi/GDAL/


GDAL - Geospatial Data Abstraction Library

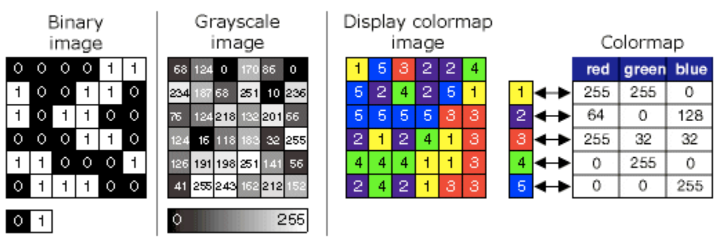
Select language: [English][Russian][Portuguese]

GDAL is a translator library for raster geospatial data formats that is released under an X/MIT style Open Source license by the Open Source Geospatial Foundation. As a library, it presents a single abstract data model to the calling application for all supported formats. It also comes with a variety of useful commandline utilities for data translation and processing. The NEWS

Access to single-band raster data

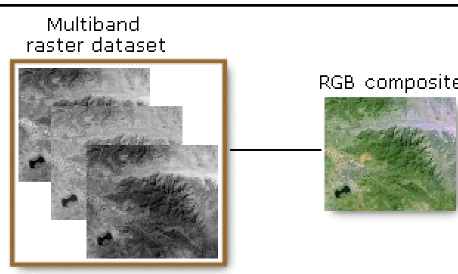


- DEM (each cell with only one value)
- Areal photographs (e.g., 8 bit)
- **Binary** images (parcel maps, query results)
- **Color map** images (land cover maps)

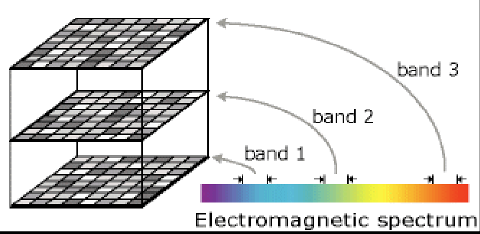


Show DEM & access in Class08_rstobj

Access multiple raster bands

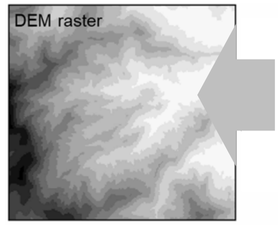


- **Satellite** imagery
- N single 2D matrices of cell values
- **Spatially coincident matrices** of cell values (same area)
- **Band** = Segment of the **electromagnetic** spectrum
- Several values at each **cell location**



Show Landsat & access in Class08_rstobj

Write out a new image object



2D numpy array

645	650	654	658	653	648
664	666	670	672	668	659
678	682	684	693	689	680
703	708	714	721	719	716
728	732	738	744	745	732
730	739	744	749	748	735

- **Write out** the **Array values** into a **new image** object
- OR: Individual resulting arrays into band objects
- **Use retrieved** raster **properties** (projection, cell size, bit depth, …) for output raster layer
- Different tools/ approaches: numpy w/ gdal, arcpy

Show output in Class08_rstobj

Critical to raster analysis…

Part I: Data access: Read, Write raster objects

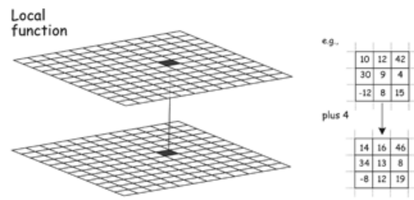
Part II: Developing local Map Algebra functions

Part III: Developing neighborhood (e.g., focal) Map Algebra functions

Part II:
Local Map Algebra functions

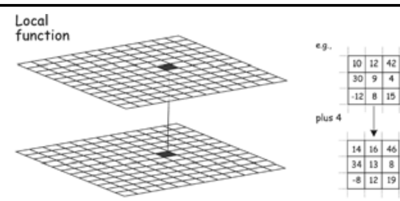
- Once raster is loaded, retrieve **individual cell values**
- Establish **cell-to-cell** relationship
- **Map Algebra** and raster **overlay**
- Raster **queries**
- Write **output** arrays

scipy: https://scipy.org/

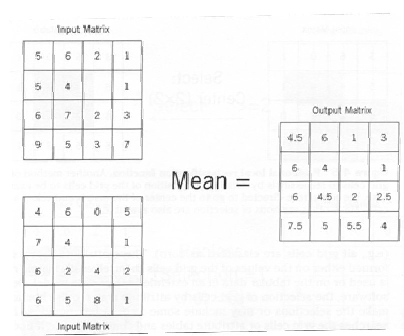


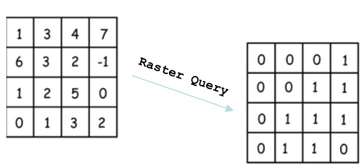
Class08_01_numpy; class08_02_rstobj

Operators in local functions



- **Uniform** cell size presumed for cell-by-cell analysis
- Mathematical/arithmetic
- **Logical**
- **Reclassification**
- **Nested**



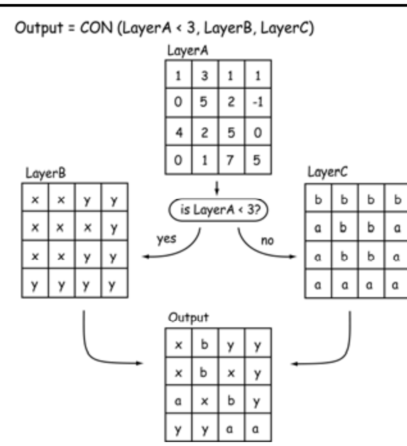


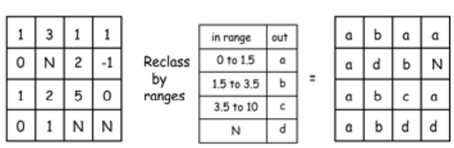
Numpy

add (+)	subtract (-)	multiply (*)	divide (/), divide_safe
remainder (%)	power (**)	*arccos*	*arccosh*
arcsin	*arcsinh*	*arctan*	*arctanh*
cos	*cosh*	*tan*	*tanh*
log10	*sin*	*sinh*	*sqrt*
absolute	*fabs*	*floor*	*ceil*
fmod	*exp*	*log*	*conjugate*
maximum	minimum		
greater (>)	equal (==)	not_equal (!=)	
greater_equal (>=)	less (<)	less_equal (<=)	
logical_or (or)	logical_xor	logical_not (not)	logical_and (and)
bitwise_or (|)	bitwise_xor (^)	bitwise_not (~)	bitwise_and (&)

Operators in local functions

- Classification/ Reclassification: Table/ Nested
- Raster clip





RECALL:
Exercise Class 07 – Raster Map Algebra in numpy
How to write a local raster query

You will practice how to do map algebra in 2-dimensional numpy arrays and create outputs that can later be written into raster data objects.
You will find an example for a numpy array (direct assignment) in the _student file. Your task is to solve the following task:

****

Select all locations with values greater than 2 AND less or equal that 5.

Create a Boolean grid as output (1/0).

Create a second grid that carries the original values where this above condition is true.

Solve this problem in a counted for loop and using numpy functions.

****

```
import numpy
myArray1 = numpy.array(([4,6,4,7,2],
                        [5,8,4,9,7],
                        [6,8,3,4,2],
                        [9,5,8,6,7],
                        [5,2,7,9,1],
                        [7,4,2,6,7]))
myArray1a = numpy.zeros(myArray1.shape).astype(float)

# get started by accessing these arrays…
```

1	3	4	7
6	3	2	-1
1	2	5	0
0	1	3	2

Raster Query

0	0	0	1
0	0	1	1
0	1	1	1
0	1	1	0

Critical to raster analysis…

Part I: Data access: Read, Write raster objects

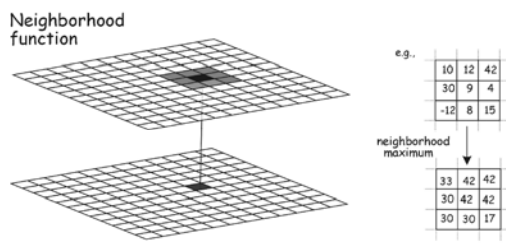
Part II: Developing local Map Algebra functions

Part III: Developing neighborhood (e.g., focal) Map Algebra functions

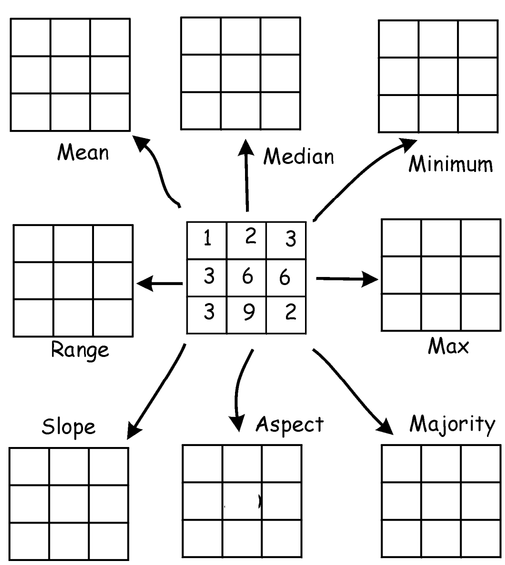
Part III:
Focal Map Algebra & convolution

- Everyday needs in raster GIS (slope, aspect)
- Neighborhoods uniform
- Analysis extent is determined by predefined neighborhood
- Moving windows
- Margin erosion

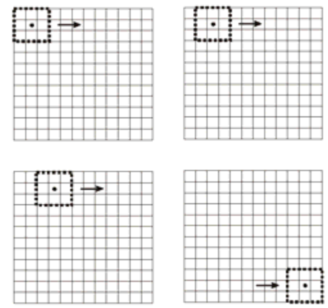
scipy: https://scipy.org/



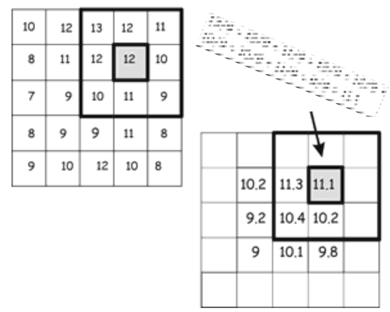
Focal functions - examples



Moving Windows



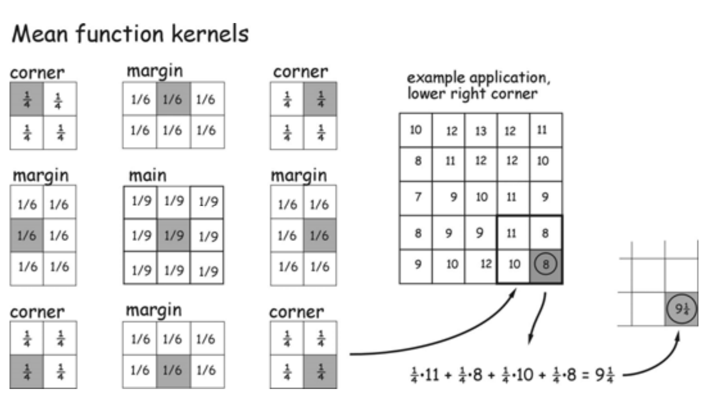
- Positioned over the input raster
- Define the input for an **operation** to be applied
- Result associated with **center** and written to the output.
- Window “**moves**” to the next location…
- Different **neighborhoods**…
- What about the **margins** of the output grid?



Class08_03_focal – Min & Sum

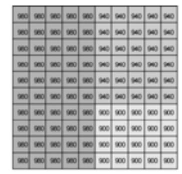
Margin Erosion

- **Enlargement** of study areas
- **MW-** and **Kernel modification** at corners and edges

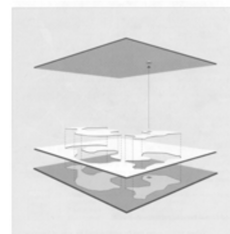


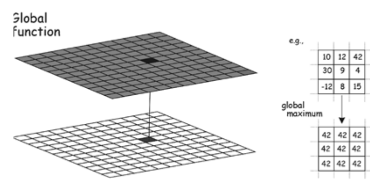
Class08_03_focal – Margins

More customized approaches to raster analysis



- Other focal functions:
  - Custom functions for **boundary** detection
  - Filters
  - Edge detection
  - Focal analysis for **directionality**
- Zonal statistics functions
- Global functions:
  - **Distance** calculations
- Morphologie
- Time series analysis, etc.





Summary

- We discussed the most important elements required for successful raster analysis in Python
- **Data access**, write raster data into arrays, and **convert** arrays back to raster data
- How to develop **local Map Algebra** functions and raster queries
- How to develop **neighborhood** (e.g., focal) Map Algebra functions

Next time

- We will look at more complex approaches of map algebra
- Identify certain boundary conditions
- Distance functions
- Morphology/ patch metrics
  - Compute patches of land cover classes
  - Land cover change